Enabling a parallel shared filesystem on Alces Flight clusters

High performance HPC filesystems on AWS

Alces Flight - 2016.3

Mark Titorenko - Alces Flight Ltd

October 2016

Revision number	Release date	Release notes	Author(s)
1.0	October 2016	Initial release	Mark Titorenko

CONTENTS

Overview

High Performance Computing (HPC) clusters provide an excellent mechanism for processing large numbers of compute jobs in a short amount of time. Clusters allow massive flexibility for customers to design and build their own environments, using either on-premises hardware or public cloud platforms like Amazon Web Services (AWS). Alces Flight has been designed to be able to utilize many different platforms to deliver a complete HPC experience, leveraging the benefits of each such as high-performance on-premises Infiniband interconnects, or the cost-effective worldwide spot market on AWS.

Many large, on-premises compute facilities also include a large, shared data storage facility in order to cater for multiple different workloads running on the HPC cluster. For clusters that have many thousands of compute cores, the attached filesystem may need to be many hundreds of terabytes in size, and deliver data at high speeds to the attached compute nodes. The challenge for sites investing in such facilities is in determining how much capacity and performance they need to deploy – an on-premises cluster might run many different workloads, all of which have different I/O profiles. Facility managers are often forced to deliver a “best guess” storage platform, which can result in future bottlenecks, or under-utilisation of expensive storage hardware.

This whitepaper discusses on-demand deployment of a parallel filesystem to support an Alces Flight Solo compute cluster. We demonstrate that a stable, high-performance and scalable filesystem can quickly and easily be instantiated to support a wide range of workloads, and provide measured performance figures and cost estimates for such a solution.

Intended audience

This white paper is primarily intended for use by users and administrators who are already utilizing HPC clusters for their research and scientific computing requirements. The paper is not a detailed configuration guide - but does advise best practices for particular use-cases and scenarios.

This paper assumes that you have an existing Alces Flight Compute environment set up, or are able to set up an Alces Flight Compute environment in order to support the steps described. For further information on launching your own Flight cluster, including further whitepapers that describe the launch mechanism and customization steps, visit our website at the following URL:

http://alces-flight.com

What is a parallel filesystem?

The majority of Linux HPC workloads still rely on using a POSIX filesystem to access data. Where many nodes work on a single job in parallel, it's common for all the nodes to need to access the same data files at the same time. At the end of the job, some of the nodes typically write out the results of the job to a file, and well-designed parallel HPC jobs will allow check-pointing while a job is running – allowing all the nodes to save their current state to a file on disk.

A shared filesystem can accommodate most of these requirements – protocols such as NFS are ideal for simply sharing out a storage volume from a single NFS server to be mounted on all your compute nodes. This is the default mechanism used for an Alces Flight cluster launched from AWS Marketplace, and is both cost-effective and sufficiently high performance for generic workloads.

However – as many HPC users will know, not all jobs have the same characteristics when it comes to accessing and storing data. As jobs become more data hungry, and are able to run over increasing numbers of compute nodes, a single-NFS server can become saturated, potentially becoming a bottleneck, and causing jobs to run more slowly.

For workloads requiring more storage throughput, the use of a parallel filesystem is recommended. Unlike the single-fileserver model used by the standard NFS protocol, parallel filesystems can support multiple storage servers working together, delivering increased performance for your job. Many different filesystems are available, all with their own performance characteristics – in general, they all efficiently use as much storage hardware as made available to them, delivering aggregate filesystem bandwidth made up of all the component servers.

What performance is available?

If a parallel filesystem is doing its job well, then the available throughput can exceed 90% of the sum of all the available storage server hardware. For example, if four servers are capable of storing data sent to them over a network at 1GB/sec each, then a parallel filesystem configured using those servers may be able to store data at up to 3.6GB/sec. There are many factors involved when determining filesystem performance, and plenty of reasons that your job might not obtain the top-level performance you want. Some of those factors include:

- Network bottlenecks; your storage servers may be capable of top-end performance, but careful network design is required to maximise overall throughput.
- Number of client systems; you will typically need to balance the number of clients and storage servers to achieve best performance.
- Type of data access; most parallel filesystems treat file metadata (e.g. file names, directory structure, permissions, etc.) differently to the file contents. You may need to use different types of servers for each portion of your files, which can lead to different throughput when performing different file operations (e.g. read, write, copy, mkdir, etc.)

- Data resilience; some parallel filesystems support advanced features including file replication, snapshots, tiering and data migration which can effect performance.

One of the issues with these factors is that very few HPC users have the opportunity to simply try different configurations and filesystems to find out which works well for their jobs.

How do I keep my data safe?

Parallel filesystems can be a great way of cost-effectively delivering very high performance storage resources for a compute cluster using relatively modest components. If your job is using expensive compute resources, then it can be worth investing in high-performance storage to ensure that you're making best use of the compute cycles available. However – as a parallel filesystem essentially stores sections of data on multiple storage servers, your files can potentially be more susceptible to loss if an individual storage server was to fail.

Some filesystems have built-in mechanisms to help protect against loss, with examples including the generation of parity data and replication / snapshot features to allow backup copies of your data to be made. However – these features are often expensive to license and can often affect performance, reducing the overall usefulness of the parallel filesystem.

As a result, the highest performing storage tier for an HPC cluster is often known as a "scratch" filesystem – a term that helps to remind users of its status as a temporary storage area which is available while jobs are running, but isn't intended for long-term file storage. Many sites do not apply capacity quotas to scratch storage, and often routinely reformat the filesystem both to keep performance at a maximum, and to discourage long-term data from being stored there.

To keep files safe, we recommend that users store persistent data in an object-storage service, or on a fully-managed filesystem with a regular backup and business-continuity policy. For more information on using the tools included with your Alces Flight cluster to access object storage systems, see the documentation link below, or see our website for further whitepapers which discuss long-term data storage in more detail:

http://docs.alces-flight.com/en/stable/databasics/data_basics.html
http://alces-flight.com

Choosing a Parallel Filesystem

There are many different considerations when choosing a parallel filesystem for your cluster. Overall performance is obviously an important factor, although users should be aware that it's often possible to tune most filesystems to perform well for synthetic benchmarks, but more difficult to obtain overall good performance across a wide range of applications. In a persistent on-premises environment, administrators may choose a filesystem for its data-protection features, or proven ability to scale to a certain capacity or performance level. However – advanced features often come at a price, and licensing costs can quickly become significant as a filesystem grows.

A good place to start is to choose a filesystem which is well supported on the platform you intend to use for your cluster, and provides you with the ability to try it out in a test environment before choosing which support mechanism will work best for you. A full discussion of all the available parallel filesystem technologies is outside the scope of this whitepaper, but we encourage users to do their own research and evaluate a number of options before choosing their preferred solution.

We have chosen the BeeGFS parallel filesystem for the tests performed in this whitepaper. BeeGFS is an open-source, POSIX compatible parallel filesystem that provides both metadata and object-data scalability over a number of storage servers. It is well supported for Linux operating systems running on AWS, has no specific hardware requirements and is available with commercial filesystem support from ThinkParQ for production environments. The filesystem has performed well in our testing, proving to be both reliable and scalable across a wide range of storage server types.

Parallel Filesystem Architecture

The BeeGFS parallel filesystem has a number of different storage service components that are used during normal operation; they are:

- **Metadata service**; the metadata service is responsible for handling information about the files stored on the BeeGFS filesystem. Typically this information includes file names, permissions, ownerships and the directory structure that files are stored in.
- **Storage service**; the storage service is responsible handling the contents of files stored on the BeeGFS filesystem.
- **Client service**; each cluster compute or login node which accesses the BeeGFS filesystem runs the client service, which is responsible for communicating with servers running the other filesystem services.
- **Management services**; additional services provide a central registry and watchdog functionality for the filesystem, as well as management and status monitoring.

In order to configure and use a BeeGFS parallel filesystem, an administrator must find somewhere appropriate to run each of the filesystem services. The choice of server or instance types for each service governs the performance, capacity and availability of the resulting BeeGFS filesystem.

Choosing your filesystem servers

The Alces Flight and BeeGFS software products are specifically designed to be platform independent. This allows users the flexibility to deploy their software in a wide range of different scenarios to support their research and compute requirements, without being limited by compatibility restrictions. One of the major benefits of open-source software is that everyone is free to experiment with different platforms to discover what works best for them. For individuals, institutions and companies that want to pay for a faster resolution without the research – both Alces and ThinkParQ can supply consultancy services to design and implement appropriate configurations for specific workloads.

In researching this whitepaper, we have made the following choices of cluster and filesystem server, based on available AWS instance types:

- **BeeGFS metadata and management service host**
  - 1 x c3.large instance
- **BeeGFS storage service hosts**
  - *Option 1*: 8 x c3.8xlarge instances (each with 640GB of instance SSD disks)
  - *Option 2*: 2 x i2.8xlarge instances (each with 6.4TB of instance SSD disks)
  - *Option 3*: 2 x d2.8xlarge instances (each with 48TB of instance magnetic disks)
- **BeeGFS client hosts**
  - Alces Flight Solo Professional cluster (version 2016.3r2)
    - Login node – c4.8xlarge instance
    - Compute nodes – c4.8xlarge instances

As network performance can be a bottleneck for many parallel filesystem deployments, our selections have generally featured instance types with 10-gigabit Ethernet network connections. The exception to this recommendation is the instance hosting metadata services, which requires a host capable of a high network transaction rate, but not necessarily high overall network bandwidth.

We recommend that AWS on-demand instance types are used for all filesystem servers, and the cluster login node (the default for Flight clusters). Users can choose between on-demand or spot compute node instances, depending on the available budget and ability to restart any aborted batch jobs if outbid on the spot market. Onboard instance disks were used as backing storage for the BeeGFS filesystem - a wide range volumes types are also available via the AWS Enterprise Block Storage (EBS) service, and are compatible with BeeGFS. Flight cluster compute and login nodes use EBS volumes for system disks, with a shared NFS filesystem deployed from the login node and mounted on all compute nodes.

Three filesystem options have been tested in order to demonstrate the flexibility available to users to choose between parallel filesystem performance, capacity and availability options. These options provide the following features:

Filesystem option	Storage service instance features				
	Instance quantity	Total server network bandwidth	Formatted BeeGFS filesystem capacity	Total disks	Total cost ($/hr)
Option 1	8	80Gb	5TB	16 x SSD	$15.41
Option 2	2	20Gb	12TB	16 x SSD	$15.12
Option 3	2	20Gb	88TB	48 x magnetic	$11.88

Costs shown include one c3.large on-demand instance ($0.12 per hour) to host the filesystem metadata and management services, and relate to AWS on-demand instance costs in the EU-west region during October 2016 when testing was performed. For simplicity, pricing for the Flight Compute cluster is not included (as costs depend on the instance types and pricing model selected), and do not include data costs for downloading data from AWS, as some customers are exempt from these charges. Note that there are usually no data charges for storing/retrieving data on your parallel filesystem from a Flight Compute cluster, as they are typically deployed in the same AWS region and availability zone.

Deploying an Alces Flight cluster on AWS

As well as being an excellent tool for production HPC workloads, Alces Flight Solo also provides an ideal environment for testing different configurations due to its ability to rapidly deploy in different configurations, with a suite of benchmarking tools readily available. The Community Edition of Flight Solo is available as a free-to-use environment for users, with Flight Solo Professional adding the option of commercial cluster support, and helpful additional features.

All testing was performed using a Flight Solo cluster deployed from AWS Marketplace. Flight automatically configures a new VPC at launch time, ensures all instances are contained in placement group for best network performance, and includes high-performance network drivers for supported AWS instance types. Once launched, the benchmarking software depot from Alces Gridware was installed, using the following command:

```
# alces gridware depot install benchmark
```

For full instructions on how to launch your own Alces Flight cluster on AWS, follow the instructions in our documentation at the following URL:

http://docs.alces-flight.com/en/stable/launch-aws/launching_on_aws.html

Adding a BeeGFS filesystem to your cluster

There are many different considerations when configuring a parallel filesystem for your own cluster compute environment. As well as the number and type of instances to use for the filesystem services, users need to choose backing-filesystem types, where to mount the filesystem, and give careful thought to maximizing resiliency and throughput for cluster compute nodes. The BeeGFS website includes a wealth of useful information to help guide users in creating their own filesystems, and is available at the following URL:

http://www.beegfs.com/content/documentation/

For testing purposes, Alces have created two automated methods that quickly configure a BeeGFS filesystem to attach to your Flight Solo cluster. Although these templates make a number of important decisions for you, these methods provide new users with a quick and simple way to get up and running quickly with a parallel filesystem on AWS.

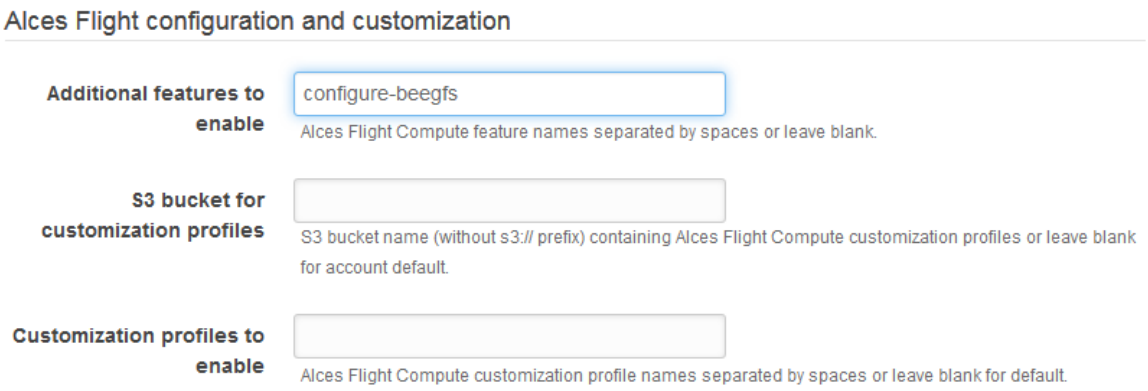
Method 1: Launching a Flight Solo cluster from AWS Marketplace

This method is suitable for users who want to launch an Alces Flight Solo cluster from the AWS Marketplace, and choose to attach a parallel filesystem at a later time. It is compatible with both Community and Professional editions of Flight Solo version 2016.3 and above. Follow the steps below to launch your cluster, and create and attach a BeeGFS parallel filesystem:

1. Subscribe to either the Community or Professional edition of Alces Flight Solo in AWS Marketplace, depending on the desired features and support level.

2. Follow the instructions at the URL below to configure and launch your cluster:
   http://docs.alces-flight.com/en/stable/launch-aws/launching_on_aws.html

3. When answering the CloudFormation template questions, specify "**`configure-beegfs`**" in the *Additional Features to enable* box, as shown in the screenshot below:



4. Once your Flight cluster is launched, use the "alces about identify" command to collect the master node IP address, cluster UUID and security token.

5. Launch the CloudFormation template included in appendix A of this whitepaper, entering the information collected in step 4 above. Select your Flight cluster VPC, subnet and placement group for best performance.

6. Once launched, your filesystem will be created and mounted as **`/sharedscratch`** on the cluster login and compute nodes.

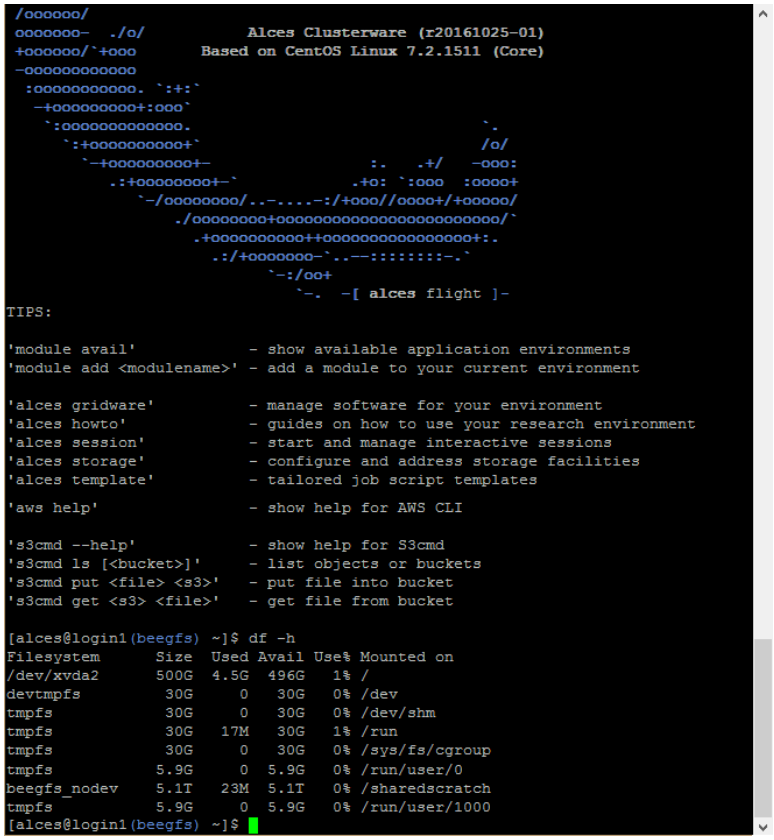
Method 2: Launching a Flight Solo cluster and BeeGFS filesystem from a single template

This method is suitable for users who want to launch a new Alces Flight Solo cluster that has a BeeGFS parallel filesystem fully integrated with it. A single AWS CloudFormation template has been included in appendix A to this whitepaper which performs all necessary functions to create the desired environment. As well as choices for the Flight cluster login and compute nodes, users can also choose the instance type and quantity for BeeGFS metadata and storage service hosts. The filesystem is automatically mounted on the **`/sharedscratch`** mount-point on all cluster login and compute nodes.

The template provided is intended to be used to demonstrate functionality, and to allow users to begin testing their own workloads on AWS. Users can also modify the template to allow them to achieve different functionality, and extend the solution to include different instance types.

Using a BeeGFS filesystem on a Flight cluster

Once launched, your BeeGFS parallel filesystem will automatically be configured and mounted at **`/sharedscratch`** on your cluster login and compute nodes. Users can store data in the filesystem in the same way as files and directories are stored in their NFS-shared **`/home`** directory:



Users are free to configure and modify their BeeGFS filesystem as required to suit their environment; documentation is provided on the BeeGFS website, at the following URL:

http://www.beegfs.com/content/documentation/

Once testing is completed, users should ensure that any data they want to keep has been safely transferred off the cluster filesystems to an appropriate object storage or protected file storage system before terminating their AWS Cloudformation stack. Deleting the stack will release all AWS resources used by both the Alces Flight cluster and the BeeGFS filesystem – ensuring that no further charges are made by AWS for the environment.

Benchmarking the parallel filesystem

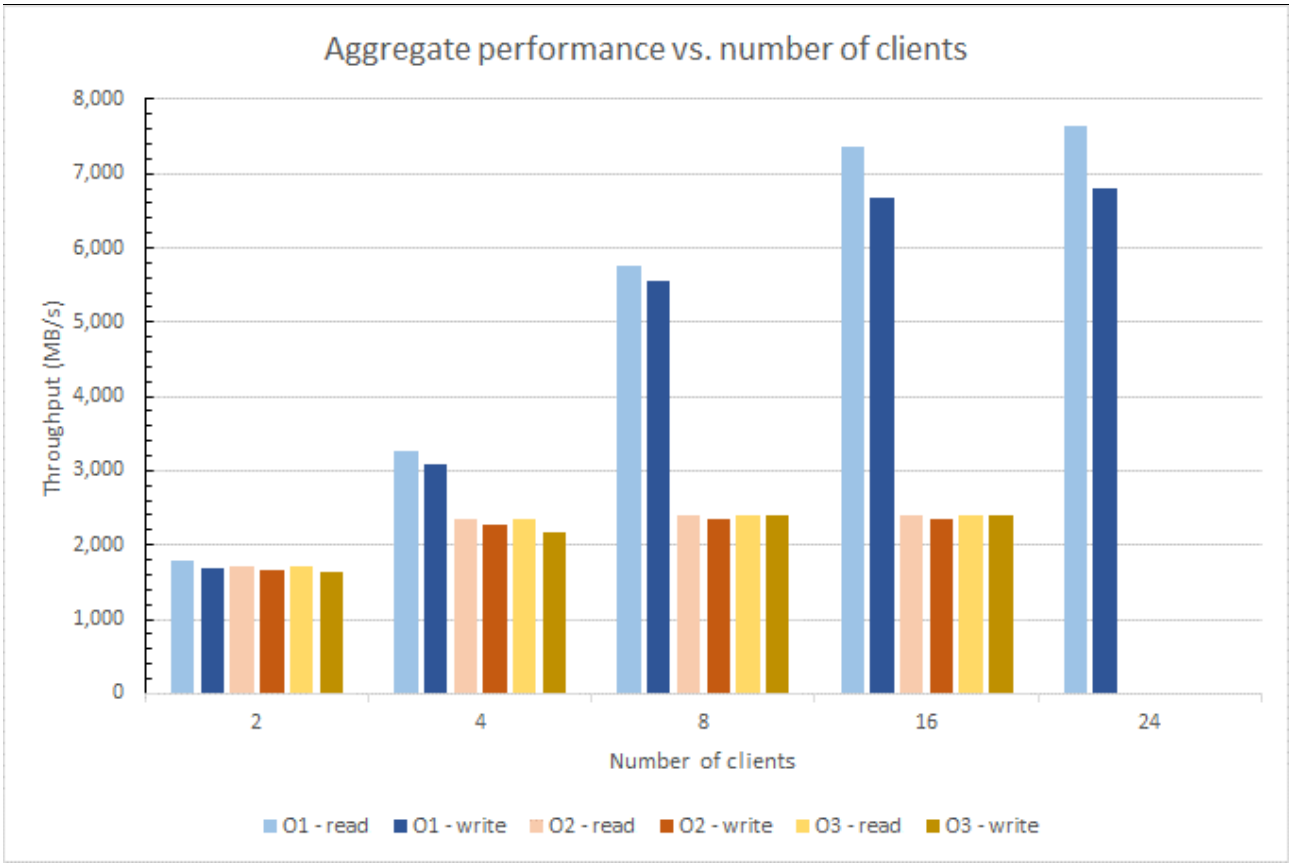
To get an overview of the performance characteristics of your parallel filesystem, it is important to measure aggregate filesystem performance by using a number of clients working simultaneously. Single-threaded and single-client workloads typically cannot saturate a parallel filesystem due to the limitations of an individual compute or login node. The **`iozone`** benchmark incorporates a multiple-client testing mode which can be used to demonstrate aggregate performance of a filesystem across a number of compute nodes. Full usage documentation for the utility is provided at the *Iozone* website, at the following URL:

http://www.iozone.org/

All three filesystems were tested with up to 24 x c4.8xlarge compute nodes, using sequential read and sequential write tests to simulate a large file workload. Each client read/wrote one file each, with BeeGFS automatically determining file placement on the storage service hosts. A record size of 1MB was used for each test, with each client reading/writing 10GB of data, with filesystem caches being cleared on each host before each new test. A full summary of the test parameters used, along with the BeeGFS and Alces Flight versions tested is provided in appendix B.

A summary of the results obtained from the three filesystem options are shown below;

	Aggregate sequential filesystem performance (MB/s)									
Number of clients	**2**		**4**		**8**		**16**		**24**	
Filesystem option	*Write*	*Read*	*Write*	*Read*	*Write*	*Read*	*Write*	*Read*	*Write*	*Read*
Option 1 – 8 x c3	1,699	1,789	3,094	3,267	5,543	5,755	6,672	7,369	6,810	7,646
Option 2 – 2 x i2	1,650	1,723	2,267	2,351	2,342	2,404	2,341	2,404	-	-
Option 3 – 2 x d2	1,648	1,704	2,179	2,337	2,404	2,405	2,402	2,404	-	-



The results of our performance tests demonstrate that throughput from the parallel filesystem scales with the number of storage servers and clients that are used in your cluster. For best performance when using 10-gigabit Ethernet networking on AWS, an optimal ratio is two clients for each storage service host should be used. Up to this limit, excellent scaling was demonstrated by the filesystem created with eight storage service hosts, highlighting that the BeeGFS parallel filesystem can deliver scalable performance when deployed across multiple servers.

We anticipate that for real-world deployments, it would be important to balance the instance types used to create the BeeGFS filesystem with the requirements of your workload. Although our tests demonstrated similar throughput performance between SSD-backed and magnetic-disk backed instances, there are likely to be significant differences in random file and metadata operation performance which will be influenced by the configuration of the storage service hosts.

Making sense of it all

Our testing has demonstrated the sequential performance that can be obtained by deploying a BeeGFS parallel filesystem on AWS to support an Alces Flight Solo compute cluster. Ultimately, cluster users need to review the available performance and capacity needed for their jobs, and choose a configuration that suits the way their cluster will be used. As well as performance, both available capacity and cost should be considered, as demonstrated by the statistics delivered by our three test filesystems:

Filesystem	Number of OSS	Seq read (GB/s)	Seq write (GB/sec)	Capacity (TB)	Total cost ($/hr)	Cost per TB ($/hr)	Cost per GB/sec write ($/hr)
Option 1	8 x c3	7.65	6.81	5	$15.41	$3.08	$2.06
Option 2	2 x i2	2.40	2.34	12	$15.12	$1.26	$6.44
Option 3	2 x d2	2.41	2.40	88	$11.88	$0.14	$5.06

For more information, we encourage users to launch their own environments to evaluate using a parallel filesystem for their own workloads. Further resources are provided at the URLs below which detail how to use Alces Flight Solo clusters, installing and configuring BeeGFS, and using the AWS platform for your scientific computing requirements.

https://docs.alces-flight.com/
http://www.beegfs.com/
https://aws.amazon.com/ebs/

Appendix A: AWS CloudFormation templates

The following AWS CloudFormation templates can be used to launch a BeeGFS filesystem in a configuration suitable for use with an Alces Flight Solo cluster. Before using the templates, please ensure that you have subscribed to the Alces Flight Solo Community Edition product for free at the AWS Marketplace, using the following URL:

http://tiny.cc/alcesflightmp

- **Template 1**: Launches a BeeGFS filesystem environment in the EU-west or US-east AWS regions, suitable for attaching to an Alces Flight Solo compute cluster which is already running:

  https://s3-eu-west-1.amazonaws.com/flight-aws-marketplace/2016.3/dev.solo.integrated-storage.json

- **Template 2**: Launches an Alces Flight Solo Community Edition cluster, with a BeeGFS shared parallel filesystem in the EU-west or US-east AWS regions:

  https://s3-eu-west-1.amazonaws.com/flight-aws-marketplace/2016.3/dev.solo.community-integrated-cluster%2Bstorage.json

Appendix B: Test environment

This section has been included as a reference to allow users to understand the original test environment, to allow results to be repeated and validated.

- ***Environment***:
  - Alces Flight version: **Solo Professional 2016.3r2**
    - Clusterware release: **r2016.3.2**
    - Redhat release: **CentOS Linux release 7.2.1511 (Core)**
    - Kernel revision: **3.10.0-327.36.3.el7.x86_64**
  - AWS AMI-ID: **ami-0901497a**
  - AWS pricing: **Valid for EU-west-1 region at time of publishing (October 2016)**
  - Instance tuning settings:
    - aTHP: **disabled**
    - clocksource: **TSC**
    - AWS AZ: **eu-west zone-B**
    - AWS network: **enhanced network driver with placement group**
    - Hyperthreading: **disabled**
    - Flight node system disk EBS type: **gp2**
    - jumbo frames: **default (unset) MTU 9001**

- ***Test applications***
  - BeeGFS
    - Version: **2015.03 (rhel7)**
    - Storage server backing filesystem type: **XFS**
      - FS creation options (MDS): **"version=2,su=128k -isize=512"**
      - FS creation options (OSS): **defaults**
      - FS mount options (MDS): **"-onoatime,nodiratime,nobarrier"**
      - FS mount options (OSS): **"-onoatime,nodiratime,logbufs=8, logbsize=256k,largeio,inode64,swalloc,allocsize=131072k"**
    - Storage server kernel scheduler: **deadline**
    -
  - Iozone:
    - Alces gridware package version: **apps/iozone/3.420/gcc-4.8.5**
    - Write test parameters**: "-i 0 -t $CLIENTNUM -+m clientlist-24 -r 1M -s 10G –w"**
    - Read test parameters**: "-i 1 -t $CLIENTNUM -+m clientlist-24 -r 1M -s 10G –w"**
    - Cache flush method between runs: **"echo 3 > /proc/sys/vm/drop_caches"**