Optimizing Electronic Design Automation (EDA) Workflows on AWS

October 2017

Notices

Contents

Introduction 1

EDA Overview 1

Benefits of the AWS Cloud 2

Improved Productivity 2

High Availability and Durability 3

Matching Compute Resources to Requirements 3

Accelerated Upgrade Cycle 4

Selecting Your Instance and Storage Solution 5

Selecting Your Instance 5

Selecting Your Storage Solution 8

Optimizing EDA Software on AWS 15

Amazon EC2 Instance Types 15

Operating System Optimization 16

Enhanced Networking 20

Amazon EBS Optimization 20

Kernel Virtual Memory 20

Paths for Migrating EDA Workflows to the AWS Cloud 21

Cloud-Optimized Traditional Architecture 21

Data Access and Transfer 23

Licensing 23

Security and Governance in the AWS Cloud 24

Isolated Environments for Data Protection and Sovereignty 24

User Authentication 25

Network 26

Data Storage and Transfer 27

Governance and Monitoring 29

Contributors 30
Document Revisions 31

Abstract

Organizations in the electronic design automation (EDA) space can significantly accelerate their product development lifecycle by taking advantage of the near infinite compute, storage, and resources available on AWS. This whitepaper presents an overview of the EDA workflow, and the specific AWS architectural components to maximize EDA workflows.

Introduction

The workflows, applications, and methods used for the design and verification of semiconductors, integrated circuits, and printed circuit boards have been largely unchanged since the invention of computer-aided engineering (CAE) and electronic design automation (EDA) software. However, as electronics systems and integrated circuits have become more complex with smaller geometries, the computing power and infrastructure requirements to design, test, validate, and build these systems have grown significantly. CAE, EDA, and emerging workloads such as computational lithography and metrology have driven the need for massive scale computing and data management in next-generation electronic products.

In the semiconductor and electronics sector, a large portion of the overall design time is spent verifying components. For example, in the characterization of intellectual property (IP) cores, and for full-chip functional and timing verifications. EDA support organizations—the specialized teams that provide infrastructure support for semiconductor companies—must invest in increasingly large server farms and high-performance storage systems in order to allow for high quality and fast turnaround of semiconductor test and validation. For example, to enable rapid completion of hardware regression testing or to recharacterize design IP for new and updated fabrication technologies.

By using AWS, semiconductor companies are now able to take advantage of more rapid, flexible deployment of CAE and EDA infrastructure. The large AWS hardware fleet is available on a dynamic, as-needed basis without the significant upfront capital expenditure that is typically required for performance-critical workloads.

You can improve the throughput of these workflows by doing application-specific performance tuning. This whitepaper can help you understand how to efficiently manage data using the capabilities of AWS.

EDA Overview

EDA is a workflow and a supporting set of tools that enable the efficient design of microelectronics, and in particular semiconductor integrated circuits (ICs). Semiconductor design and verification relies on a set of commercial or open-source tools, collectively referred to EDA software, that expedites and reduces time to silicon tape-out and fabrication. EDA is a highly iterative process that can take from months to years to produce a single integrated circuit.

The increasing complexity of integrated circuits has resulted in an increased use of preconfigured or semi-customized hardware components, collectively known as intellectual property (IP) cores. These cores are either designed in-house by a semiconductor company, or purchased from a third-party IP vender, and requires EDA workflows for IP core design, verification, and characterization. These IP cores are used in combination, along with other customer-designed components, to create a complete integrated circuit or system-on-chip (SoC), which itself requires large amounts of EDA processing for full-chip verification—including modeling (that is, simulating) all of the components on the chip. This modeling, which includes initial verification (for example, using techniques such as formal verification and design-rule-check or DRC), is known as the *front-end design*. The physical implementation, which includes synthesis, place and route, timing analysis, and final verification, is known as the *back-end design*. When the back-end design is complete, a file is produced in GDSII format. The production of this file is known *tapeout* (for historical reasons). When completed, the file is sent to a fabrication facility, which might or might not be operated by the semiconductor company, where the actual component (a silicon wafer) is manufactured. From there, the integrated circuit is tested, assembled, and packaged onto a board or other system.

Benefits of the AWS Cloud

Before discussing the specifics of moving EDA workloads to AWS, it's worth noting the benefits of cloud computing on the AWS Cloud.

Improved Productivity

Organizations that move to the cloud can see a dramatic improvement in development productivity. Your organization can achieve this by scaling out your compute needs to meet the demands of the jobs waiting to be processed. With AWS, you are charged per second of compute time for as many computing resources (for example, CPU cores) that you actually need to complete your work. By going wide, you can run more compute servers (that is, Amazon EC2 instances) for a shorter period of time, and pay the same amount as if you were running fewer servers for a longer period of time. For example, because the number of compute hours consumed are the same, you could complete a 48-hour regression workflow in just two hours by dynamically growing your cluster by 24X or more in order to run many thousands of pending jobs in parallel. These extreme levels of parallelism are commonplace on AWS, across a wide variety of industries and performance-critical use cases.

High Availability and Durability

Amazon EC2 is hosted in multiple locations worldwide. These locations are composed of regions and Availability Zones. Each AWS Region is a separate geographic area around the world such as Oregon, Virginia, Ireland, and Singapore. Each AWS Region where Amazon EC2 runs is designed to be completely isolated from the other regions. This achieves the greatest possible fault tolerance and stability. Resources aren't replicated across regions unless you specifically configure your services to do so.

Within each geographic region, AWS has multiple, isolated locations known as Availability Zones. Amazon EC2 provides you the ability to place resources, such as EC2 instances, and data in multiple locations using these Availability Zones. Each Availability Zone is isolated, but the Availability Zones in a region are connected through low-latency links. By taking advantage of both multiple regions and multiple Availability Zones, you can protect against failures and ensure you have enough capacity to run even your most compute intensive workflows. For more information, see AWS Global Infrastructure.[1]

Matching Compute Resources to Requirements

AWS offers many different configurations of hardware, called instance families, in order to enable customers to match their compute needs with those of their jobs. Because of this and the on-demand nature of the cloud, you can get the exact systems you need for the exact job you need to perform for only the time you need it.

Amazon EC2 instances come in many different sizes and configurations. These configurations are built to support jobs that require both large and small memory footprints, high core counts of the latest generation processors, and storage requirements from high IOPS to high throughput. By right sizing the instance to the unit of work it is best suited for, you can achieve higher EDA performance at lower overall cost. You no longer need to purchase EDA cluster hardware that is entirely configured to meet the demands of just a few of your most demanding jobs. Instead, you can choose servers and build entire clusters that are uniquely optimized for specific applications and specific stages of chip development.

For example, consider a situation where you're performing gate-level simulations for a period of just a few weeks, such as during the development of a critical IP core. In this example, you need to have a cluster of 100 machines (representing over 2,000 CPU cores) with a specific memory footprint and a specific storage configuration. With AWS, you can deploy and run this cluster, dedicated only for this task, for as long as the simulations require, and then terminate the cluster when that stage of your project is complete.

Now consider another situation in which you have multiple semiconductor design teams working in different geographic regions, each using their own locally installed EDA IT infrastructure. This geographic diversity of engineering teams has productivity benefits for modern chip design, but it can create challenges in managing large-scale EDA infrastructure (for example, to efficiently utilize globally licensed EDA software). By using the AWS Cloud to augment or replace these geographically separated IT resources, you can pool all your global EDA licenses in a smaller number of locations using scalable, on-demand clusters on AWS. As a result, you can more rapidly complete critical batch workloads such as static timing analysis, DRC, and physical verification.

Accelerated Upgrade Cycle

Another important reason to move EDA workloads to the cloud is to gain access to the latest processor, storage, and network technologies. In a typical on-premises EDA installation, you must select, configure, procure, and deploy servers and storage devices with the assumption that they remain in service for multiple years. Depending on the selected processor generation and time-of-purchase, this means that performance-critical, production EDA workloads might be running on hardware devices that are already multiple years, and multiple processor generations, out of date. When using AWS, you have the opportunity to select and deploy the latest processor generations within minutes and configure your EDA clusters to meet the unique needs of each application in your EDA workflow.

Selecting Your Instance and Storage Solution

To get started, first you need to evaluate which EC2 instance and AWS storage solution works best for your organization.

Selecting Your Instance

When running EDA software on AWS, you should choose current generation instances that feature the latest generations of Intel Xeon processors, using a few different configurations to meet the needs of each application in your overall workflow. While AWS has many different types and sizes of instances, the instance types in the compute optimized and memory optimized categories are typically best suited for EDA workloads.

Compute Optimized Instances

The compute optimized instance family features instances that have the highest clock speeds available on AWS, along with moderate amounts of RAM, typically around 4GB per physical core. The most recent generation of compute optimized instances is the C4 instance. The C4 instances feature up to 18 physical CPUs (36 threads) and up to 60 GiB of RAM. The processor used in C4 instances is an AWS-specific Intel Xeon E5-2666 v3 (Haswell) processor with a base clock speed of 2.9 GHz and the ability to turbo boost up to 3.5 GHz. The upcoming C5 instances are also an excellent choice for EDA workflows, offering up to 36 Intel Xeon Platinum processors, part of the next-generation Intel Xeon Scalable (Skylake) processors and up to 144 GiB of RAM.

These instances are best suited for applications that don't require significant memory footprints and can benefit from higher clock speeds. Quite often we see customers use these instances for synthesis and formal verification workloads, in addition to RTL regression tests, that can take advantage of the ~4 GB per core memory footprint and the high clock speeds offered by these processors.

Memory Optimized Instances

If you require larger amounts of memory, you should consider the Amazon EC2 memory optimized family of instances. While the X1 instances, with 128 threads (64 physical cores) and 1,952 GiB of RAM, looks very interesting for thread-intensive applications, AWS testing using various EDA flows shows that M4 and

R4 64 vCPU (32 physical core) instances will usually outperform the X1 instances. As with all workflows on AWS, however, it's important to test your application on a wide variety of instance types.

The r4.16xlarge is very popular choice for memory intensive workloads in the EDA sector. The r4.16xlarge instance is based on 2.3 GHz Broadwell-based processors, with up to 64 threads (32 physical cores) and 488 GiB of RAM on a single instance. R4 instances also offer our latest generation of networking, and it can achieve 25 Gbps of bi-sectional bandwidth between EC2 instances. Because of this, it's especially attractive for nodes that require high throughput to storage subsystems (for example, to sustain high IOPS to an EC2-based shared file system as discussed in the Selecting Your Storage Solution section later in this paper).

Semiconductor organizations that run back-end design tools for place, route, and static timing analysis (STA) often choose the R4 family of instances. Because of the high core counts available on the r4.16xlarge instance, clusters of r4.16xlarge instances are commonly deployed to support these critical backend batch workflows.

Hyper-Threading

Each vCPU in the C4, M4, R4, and X1 instance families is a single hyper-thread of a physical core. You can disable Intel Hyper-Threading Technology (HT Technology) if you determine that it has a detrimental impact on your application performance.

The following table below shows the number of physical cores for each popular EDA instance size.

Table 1: Instance specifications

Instance Name	vCPU Count	Physical Core Count	RAM
x1.32xlarge	128	64	1,952 GiB
x1.16xlarge	64	32	976 GiB
r4.16xlarge	64	32	488 GiB
m4.16xlarge	64	32	256 GiB
m4.10xlarge	40	20	160 GiB
c4.8xlarge	36	18	60 GiB

To see all of the vCPUs, run the following command:

```
cat /proc/cpuinfo
```

To get more specific output, run the following command:

```
egrep '(processor|model name|cpu MHz|physical id|siblings|core
id|cpu cores)' /proc/cpuinfo
```

In this output, the `processor` is the vCPU number. The `physical id` shows the processor socket ID. For any C4 instance other than the c4.8xlarge this will be zero (0). The `core id` is the physical core number. Each entry that has the same `physical id` and `core id` are hyper-threads of the same core.

Another way to view the vCPUs pairs (that is, hyper-threads) of each core is to look at the `thread_siblings_list` for each core. This will show two numbers that are the vCPUs for each core. Change the X in `cpuX` to the vCPU number that you want to view.

```
cat /sys/devices/system/cpu/cpuX/topology/thread_siblings_list
```

To disable HT Technology, edit the `/boot/grub/grub.conf` file with your editor of choice. For the first entry that begins with `kernel` (there may be more than one kernel entry—you only need to edit the first one), add `maxcpus=NUMBER` at the end of the line, where `NUMBER` is the number of actual cores in the instance size you are using. For a list of the number of cores available in each instance type and size, see Virtual Cores by Amazon EC2 and RDS DB Instance Type on the AWS website.[2] Edit the `/boot/grub/grub.conf` file as follows:

```
# created by imagebuilder
default=0
timeout=1
hiddenmenu
```

```
title Amazon Linux 2014.09 (3.14.26-24.46.amzn1.x86_64)
root (hd0,0)
kernel /boot/vmlinuz-3.14.26-24.46.amzn1.x86_64 root=LABEL=/
console=ttyS0 maxcpus=18
initrd /boot/initramfs-3.14.26-24.46.amzn1.x86_64.img
```

Save the file and exit your editor. Relaunch your instance to enable the new kernel option.

This is a setting that you should test to determine if it provides a performance boost for your application. If you typically disable HT Technology on your physical servers to achieve better performance, you should continue that practice on AWS using the previous steps.

Selecting Your Storage Solution

For EDA workloads running at scale on any infrastructure, storage can quickly become the bottleneck for pushing jobs through the queue. Traditional centralized filers serving NFS are commonly purchased from hardware vendors at significant costs in support of high EDA throughout. However, these centralized filers can quickly become a bottleneck for EDA, resulting in reduced compute performance and correspondingly higher EDA license costs. Planned or unexpected increases in EDA data, and the need to access that data across a fast-growing EDA cluster means that the filers eventually run out of storage space, or become bandwidth constrained by either the network or storage tier.

EDA applications can take advantage of the wide array of storage options available on the AWS Cloud, resulting in reduced runtimes for large batch workloads. Achieving these benefits may require some amount of EDA workflow rearchitecting, but the benefits of making these optimizations can be dramatic.

Types of Storage on AWS

Before discussing the different options for deploying EDA storage, it’s important to understand the different types of storage services available on AWS.

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with Amazon EC2 instances in the AWS Cloud. EBS volumes are attached to instances over a high-bandwidth network fabric and appear as local block storage that can be formatted with any file system on the instance itself. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes offer the consistent and low-latency performance needed to run your workloads. With Amazon EBS, you can scale your usage up or down within minutes, while paying a low price for only what you provision.

Amazon EBS is the storage that backs all modern Amazon EC2 instances (with a few exceptions) and is the foundation for creating high speed file systems on AWS. With Amazon EBS, it's possible to achieve up to 80,000 IOPS and 1,750 MB/s all from a single Amazon EC2 instance.

Instance Storage

For those rare cases where the performance of Amazon EBS is not sufficient on a single instance, Amazon EC2 instances with instance storage are available. Instance storage is storage that is physically attached to the instance accessing it. Because it's attached to the instance, it can provide significantly higher throughput or IOPS than is available through network-based storage like Amazon EBS. However, because the storage is locally attached to the instance, data on the instance store won't tolerate hardware failures and won't persist when you stop or terminate your instance. Because of this limitation, we recommend instance storage for temporary scratch space or for data storage systems with replicated storage.

The I3 instance family is well suited for EDA workloads requiring a significant amount of fast scratch storage. I3 instances are designed for the highest possible IOPS. They use NVMe-based storage devices. The largest I3 instance, i3.16xlarge, delivers 3.3 million random IOPS at 4 KB block size and up to 16 GB/s of sequential disk throughput.

Elastic File System (EFS)

As an alternative to creating a purpose built, EDA-optimized file system (using a combination of Amazon EC2 and Amazon EBS), you can use Amazon Elastic File System (EFS). Amazon EFS provides simple, scalable NFS-based file storage for use with Amazon EC2 instances in the AWS Cloud. Amazon EFS provides a simple interface that enables you to create and configure file systems quickly and easily. With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files, so your applications have the storage they need, when they need it.

Amazon EFS is designed for high availability and durability, and can deliver high throughput when deployed at scale. Every EFS file system object (that is, directory, file, and link) is redundantly stored across multiple Availability Zones. In addition, an EFS file system can be accessed concurrently from all Availability Zones in the region where it is located. However, because all Availability Zones must acknowledge file system actions (that is, create, read, update, or delete), latency can be higher than traditional file systems that don't span multiple Availability Zones. Because of this, EFS is typically only used in the EDA space for home directories and application binary storage.

Amazon S3

Amazon Simple Storage Service (Amazon S3) is object storage with a simple web service interface to store and retrieve any amount of data from anywhere on the web. It is designed to deliver 99.999999999% durability, and scale to handle millions of concurrent requests and grow past trillions of objects worldwide. Amazon S3 offers a range of storage classes designed for different use cases. These include Amazon S3 Standard for general-purpose storage of frequently accessed data, Amazon S3 Standard – IA (for infrequent access) for long-lived, but less frequently accessed data, and Amazon Glacier for long-term archival. Amazon S3 also offers configurable lifecycle policies for managing your data throughout its lifecycle.

Amazon S3 is accessed via HTTP REST requests typically through the AWS software development kits (SDKs) or through the AWS Command Line Interface (AWS CLI). You can use the AWS CLI copy data to and from Amazon S3 in the same way that you copy data to other remote file systems, using the familiar ls, cp, rm, and sync command-line operations.

For EDA workflows, we recommend that you consider Amazon S3 for your primary data storage solution to manage data uploads and downloads, and to provide high data durability. You can quickly and efficiently copy data from Amazon S3 to Amazon EC2 instances and Amazon EBS storage. For example, to populate a high performance shared file system prior to launching a large batch regression test or timing analysis. However, we recommend that you do not use Amazon S3 to directly access (read/write) individual files during a performance-critical application. The best architectures for high performance, data-intensive computing available on AWS consists of Amazon S3, Amazon EC2, Amazon EBS, and Amazon EFS to balance performance, durability, scalability, and cost for each specific application.

Enhancing Scalability with Dynamic EBS Volumes

One of the benefits of Amazon EBS storage is that your storage footprint—at the level of a single instance, or in the form of a large shared file system—can grow based on your needs. For the typical on-premises EDA cluster, IT teams are accustomed to purchasing large arrays of network attached storage, even though their initial needs are relatively small. With Amazon EBS, you can launch a volume that is large enough to fit your needs today, and then resize that volume. You can also change its performance properties in order to store more data or have increased performance as your needs change in the future.

While most organizations use manual processes to grow, you can use AWS services to build a storage system that can automatically grow based on the amount of free space available. Amazon CloudWatch is a metrics and monitoring service available on AWS. On your NFS server, you can publish a custom metric containing the amount of free space on your server. From there, you can create custom alarms when that volume gets to a specified percentage. These alarms can send email notifications to your administrators, or more interestingly, can trigger AWS Lambda, an event-driven compute service. You can configure Lambda to perform any API action, including resizing an EBS volume. When the resize operation is complete, the new space is automatically discovered by your operating system and you can trigger a grow operation to take advantage of it.

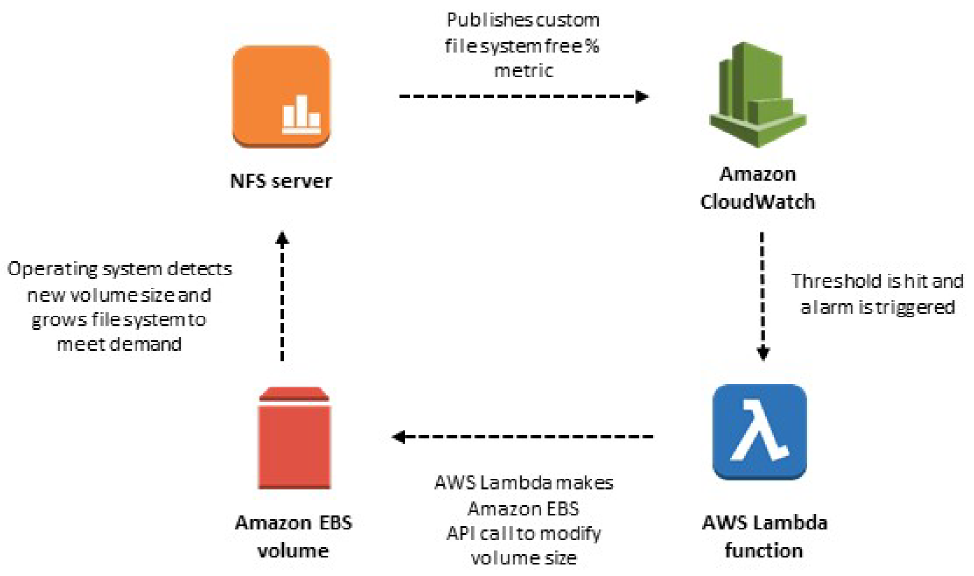
Publishes custom file system free % metric
NFS server
Amazon CloudWatch
Operating system detects new volume size and grows file system to meet demand
Threshold is hit and alarm is triggered
Amazon EBS volume
AWS Lambda makes Amazon EBS API call to modify volume size
AWS Lambda function

Figure 2: Lifecycle for automatically resizing an EBS volume

Traditional NFS File Systems

For EDA workflow migration, the first and most popular option for migrating storage to AWS is to build systems similar to your on-premises environment. This is a great first step because it enables you to migrate applications quickly without having to rearchitect your applications or workflow. With AWS, it's simple to create a storage server by launching an Amazon EC2 instance with adequate bandwidth and Amazon EBS throughput, attaching the appropriate EBS volumes, and sharing the file system to your compute nodes using NFS.

When it comes to building storage systems for the immense scale that EDA can require for large scale regression and verification tests, there are a number of approaches you can take to ensure your storage systems are able to handle the load. The largest Amazon EC2 instances support 25 Gbps of network bandwidth and up to 80,000 IPS and 1,750 MB/s to Amazon EBS. For a chart that shows the Amazon EBS throughput expected for each instance type, see Instance Types that Support EBS Optimization in the *Amazon EC2 User Guide for Linux Instances*.[3] To obtain these speeds, you need to stripe multiple EBS volumes together because each volume has its own throughput limits.[4]

For EDA workloads that require more performance in aggregate than can be provided by a single instance, you can build multiple NFS servers that are delegated to specific mount points. Typically, this means that you build servers for shared scratch, tools directories, and individual projects. By building servers in this fashion, you can right size the server and the storage allocated to it according to the demands of a specific workload. When projects are finished, you can archive the data to a low cost, long term storage solution like Amazon Glacier, and then you can delete the storage server thereby saving additional cost.

When it comes to building the storage servers, you have many options. Linux software raid (mdadm) is often a popular choice for its ubiquity and stability. However, in recent years ZFS on Linux has grown in popularity and customers in the EDA space use it for the data protection and expansion features that it provides. If you use ZFS, it's relatively simple to build a solution that pools a group of EBS volumes together to ensure higher performance of the volume, set up automatic hourly snapshots to provide for point-in-time rollbacks, and replicate data to backup servers that are in other Availability Zones in order to provide for fault tolerance.

While out of the scope of this document, if you're looking for more automated and managed solutions you should consider AWS partner storage solutions. Examples of partners that provide solutions for running high performance storage on AWS include SoftNas, WekaIO, Avere, and Netapp.

NFS Performance Considerations

Prior to setting up an NFS server, you need to enable Amazon EC2 enhanced networking. If you are using Amazon Linux, it is automatically enabled by default.

A crucial part of high performing NFS are the mount parameters on the client. For example:

```
rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2
```

A typical EFS mount command is shown following:

```
sudo mount -t nfs4 –o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2
file-system-id.efs.aws-region.amazonaws.com:/ /efs-mount-point
```

When building an NFS server on AWS, it's important to choose the correct instance size and number of EBS volumes. As you're looking at instance sizes within a single family, you'll notice that larger instances typically have more network and Amazon EBS bandwidth available to them. The largest NFS servers on AWS are often built using m4.16xlarge instances with multiple EBS volumes striped together in order to achieve the best possible performance.

Cloud Native Storage Approaches

Because of its low cost and strong scaling behaviors, Amazon S3 is well suited for EDA workflows because you can adapt them to reduce or eliminate the need for traditional shared storage systems. These optimized EDA workflows use a combination of Amazon EBS storage and Amazon S3 to achieve extreme scalability at very low costs, without being bottlenecked by traditional storage systems.

To take advantage of a solution like this, your EDA organization and your supporting IT teams might need to untangle many years of legacy tools, file system sprawl, and large numbers of symbolic links in order to understand what data you need for specific projects (or job deck) and pre-package the data along with the job that requires it. The typical first step in this approach is to separate out the static data (for example, application binaries, compilers, etc.) from dynamically changing data and IP in order to build a front-end workflow that doesn't require any shared file systems. This is an important step for optimized cloud migration, and also provides the benefit of increasing the scalability and reliability of legacy, on-premises EDA workflows.

By using this less NFS-centric approach to manage EDA storage, operating system images can be regularly updated with static assets so that they're available when the instance is launched. Then, when a job is dispatched to the instance, it can be configured to first download the dynamic data from Amazon S3 to local or Amazon EBS storage before launching the application. When complete, results are uploaded back to Amazon S3 to be aggregated and processed when all jobs are finished. This method for decoupling compute from storage provides substantial benefit, in particular for front-end RTL batch regressions.

Optimizing EDA Software on AWS

EDA software tools are critical for modern semiconductor design and verification. Increasing the performance of EDA software—measured both as a function of individual job runtimes and on the completion time for a complete set of EDA jobs—is important to reduce time-to-results/time-to-tapeout, and to optimize EDA license costs.

Choosing the right Amazon EC2 instance type and the right OS-level optimizations is critical for EDA tools to perform well. This section provides a set of recommendations that are based on actual daily use of EDA software tools on AWS—usage by AWS customers and by Amazon internal silicon design teams. The recommendations include such factors as instance type and configuration, as well as OS recommendations and other tunings for a representative set of EDA tools. These recommendations have been tested and validated internally at AWS, and with EDA customers and vendors.

Note that the information and suggestions made here are only a starting point, and may change over time as new AWS instance types are released and as EDA software tools are updated. Also note, this document focuses primarily on optimizing compute performance (software execution times), and less on storage optimization (for example, building and optimizing an NFS environment).

Amazon EC2 Instance Types

The following table highlights EDA tools and provides corresponding Amazon EC2 instance type recommendations.

Table 2: EDA tools and corresponding instance type

Tool Name	Usage	Instance Types
Formal verification	Property check and equivalence check.	c4.4xlarge c4.8xlarge
RTL Simulation – Interactive debugging	For interactive debug (for example, opening waveforms, etc.), choose an	c4.4xlarge c4.8xlarge

Tool Name	Usage	Instance Types
	instance sized for a single simulation workload and with enough memory to avoid paging.	(HT disabled)
RTL simulation – Batch mode	Batch regressions, typically run at large scale using a batch scheduler. Most RTL tools are limited by single-thread performance. Running on a larger, higher memory instance helps with performance by having higher network and Amazon EBS bandwidth. Performance can also be improved if NFS client-side caching is enabled.	r4.8xlarge 16 GB/core (HT disabled)
RTL simulation – Batch mode with multithread optimized tools.	Batch workloads run at large scale. With multithread runs, the DRAM/CPU ratio required is less than the single-threaded tools so C4 instances are a good alternative.	c4.4xlarge c4.8xlarge r4.4xlarge r4.8xlarge r4.16xlarge
RTL gate-level system simulation	Gate-level system simulations are lengthy in nature, and typically ran toward the end of the project, where shorter job run times are preferred. C4 instances provide the highest per-thread performance.	c4.4xlarge c4.8xlarge
Place and route	These applications are primarily bounded by single thread performance, and memory size. Avoid swapping to disk by choosing large memory instance types.	r4.8xlarge (244 GB) r4.16xlarge (488 GB) x1.16xlarge (976 GB) x1.32xlarge (1952 GB) (HT disabled)
State Timing Analysis	These applications are large consumers of memory, depending on the size of the input. Avoid swapping to disk by choosing large memory instance types.	r4.8xlarge (244GB) r4.16xlarge (488GB) x1.16xlarge (976GB) x1.32xlarge (1952GB) (HT Disabled)

Operating System Optimization

After you've chosen the instance types for your EDA tools, you need to customize and optimize your OS to maximize performance.

Use a Current-Generation Operating System

The recommendation in this section have been tested on the following operating systems:

- Amazon Linux[5]
- CentOS 7.x
- Red Hat Enterprise Linux 7.x
- Ubuntu 16.04

You can use previous-generation operating systems based on the 2.6 series Linux kernel with AWS. However, you should use a current-generation operating system based on the 3.10 or newer kernel to achieve the best possible performance.

Change Clocksource to TSC

By default, instances use the Xen pvclock, which is in the hypervisor. To avoid communication with the hypervisor and use the CPU clock instead, you should use `tsc` as the clock source.

To change the clock source, run the following command:

```
sudo su -c "echo tsc >
/sys/devices/system/cl*/cl*/current_clocksource"
```

To verify that the clock source is set to `tsc`, run the following command:

```
cat /sys/devices/system/cl*/cl*/current_clocksource
```

You set the clock source in the initialization scripts on the instance. You can also verify that the clocksource changed with the `dmesg` command as shown following:

```
$ dmesg | grep clocksource
...
clocksource: Switched to clocksource tsc
```

Change to Optimal Spinlock Setting

Amazon Linux and other distributions, by default, implement a paravirtualized mode of spinlock that is optimized for low-cost preempting virtual machines (VMs). This can be expensive from a performance perspective because it causes the VM to slow down when running multithreaded with locks. Because most EDA tools are not optimized for multi-core they rely heavily on spinlocks.

To disable the paravirtualized mode of spinlock, add `xen_nopvspin=1` to the kernel command line in `/boot/grub/grub.conf` and restart. Following is an example kernel command:

```
kernel /boot/vmlinuz-4.4.41-36.55.amzn1.x86_64 root=LABEL=/
console=tty1 console=ttyS0 selinux=0 xen_nopvspin=1
```

Note If your operating system is using grub2, the same setting is required, but there is a different update process for grub2.

Verify that the setting by running `dmesg` or `/proc/cmdline` kernel command line, as shown following:

```
$ dmesg | grep "Kernel command line"
[    0.000000] Kernel command line: root=LABEL=/ console=tty1
console=ttyS0 maxcpus=18 xen_nopvspin=1

$ cat /proc/cmdline
root=LABEL=/ console=tty1 console=ttyS0 maxcpus=18
xen_nopvspin=1
```

Disable Hyper-Threading

On current generation Amazon EC2 instance families (other than the T2 instance family), AWS instances have HT Technology enabled by default. To achieve maximum single thread performance for most EDA tools, you should disable hyper-threading.

You can run the following script on a Linux instance to disable HT Technology while the system is running. This can be set up to run from an init script so that it applies to any instance when you launch the instance.

```
for cpunum in $(cat
/sys/devices/system/cpu/cpu*/topology/thread_siblings_list | cut
-s -d, -f2- | tr ',' '\n' | sort -un); do
    echo 0 | sudo tee
/sys/devices/system/cpu/cpu${cpunum}/online
done
```

Or, you can disable HT Technology by setting the Linux kernel to only initialize the first set of threads by setting `maxcpus` in grub to be half of the vCPU count of the instance. After you update grub, you need to relaunch the instance to implement the change.

For example, a setting on a c4.8xlarge instance would be as follows:

```
maxcpus=18
```

When you disable HT Technology, it doesn’t change the workload density per server because these tools are demanding on DRAM size and reducing number of threads only helps as GB/core increases.

Enable Turbo Mode (Processor State)

If you are running on the largest instance size for a selected family (for example, m4.16xlarge, c4.8xlarge, or r4.16xlarge), you can take advantage of the turbo frequency boost, especially when you’ve disabled HT Technology.

Amazon Linux has turbo mode enabled by default, but other distributions may not. To ensure that turbo mode is enabled, run the following command:

```
sudo su -c "echo 0 > /sys/devices/system/cpu/intel_pstate/no_turbo"
```

For more information about Amazon EC2 instance processor states, see the Processor State Control for Your EC2 Instance in the *Amazon EC2User Guide for Linux Instances*.[6]

Enhanced Networking

Amazon's Enhanced Networking technology enables instances to communicate at 25 Gbps for current-generation instances, and 10 Gbps for previous-generation instances. Additionally, with enhanced networking, you will see reduced latency and reduced network jitter. Enhanced Networking is enabled by default on Amazon Linux. However, on CentOS or Red Hat you will need to enable it by installing the network module, and update the Enhanced Network Adapter (ENA) support attribute for the instance. For more information about Enhanced Networking, including build and install instructions, see the Enhanced Networking on Linux in the *Amazon EC2User Guide for Linux Instances*.[7]

Amazon EBS Optimization

When selecting your instance type, you should select an instance that is Amazon EBS-optimized by default. An Amazon EBS-optimized instance provides dedicated throughput to Amazon EBS which is isolated from any other network traffic and an optimized configuration stack to provide optimal Amazon EBS I/O performance. If you choose an instance that is not Amazon EBS-optimized, you can enable Amazon EBS-optimization by using `--ebs-optimized` with the `modify-instance-attribute` parameter in the AWS CLI, but additional charges may apply (cost is included with instances where Amazon EBS is optimized by default).

Kernel Virtual Memory

Typical operating system distributions are not tuned for large machines like those offered by AWS for EDA workloads. As result, out of the box configurations often have sub-optimal performance settings for kernel network buffers and storage page cache background draining. While the specific numbers may vary by instance size and applications runs, the AWS EDA team has found that these kernel configuration settings and values are a good starting point to optimize memory utilization of the instances:

```
vm.min_free_kbytes=1048576
vm.dirty_background_bytes=107374182
```

Paths for Migrating EDA Workflows to the AWS Cloud

When you begin the migration of EDA workflows to AWS, you can draw many parallels with more traditional deployments across multiple data centers. Larger organizations in the semiconductor industry typically have multiple EDA data centers that are geographically segregated because they use distributed design teams. These organizations typically choose specific workloads to run in specific locations, or replicate and synchronize data to allow for multiple sites to take the load of large-scale, global EDA workflows. If your organization uses this approach, you need to consider that the specifics around topics such as data replication, caching, and license server management depend on many internal and organizational factors.

Most of the same approaches and design decisions related to multiple data centers also apply to the cloud. With AWS, organizations can build one or more virtual data centers that mirror existing EDA data center designs. The foundational technologies that enable things like compute resources, storage servers, and user workstations are available with just a few keystrokes. However, the real power of using the AWS Cloud for EDA workloads comes from the dynamic capabilities and enormous scale provided by AWS.

Cloud-Optimized Traditional Architecture

One of the benefits of moving to AWS is that you no longer need a single cluster for every type of job that you need run across your infrastructure. Because you can configure compute resources to come and go on-demand, you can build clusters that are specific to different parts of the workflow, or even specific projects. This allows for many benefits including project-based cost allocation, right sizing compute and storage for specific purposes, and isolation of environments for security purposes.

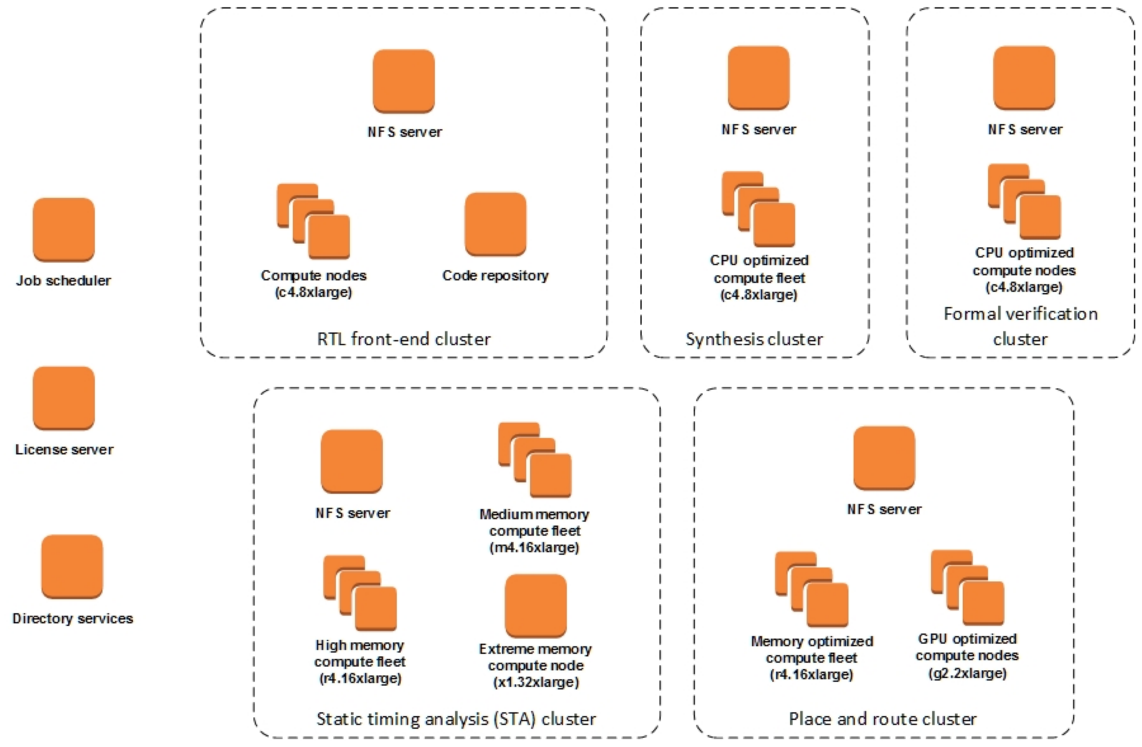
NFS server
Compute nodes (c4.8xlarge)
Code repository
RTL front-end cluster
NFS server
CPU optimized compute fleet (c4.8xlarge)
Synthesis cluster
NFS server
CPU optimized compute nodes (c4.8xlarge)
Formal verification cluster
Job scheduler
License server
Directory services
NFS server
Medium memory compute fleet (m4.16xlarge)
High memory compute fleet (r4.16xlarge)
Extreme memory compute node (x1.32xlarge)
Static timing analysis (STA) cluster
NFS server
Memory optimized compute fleet (r4.16xlarge)
GPU optimized compute nodes (g2.2xlarge)
Place and route cluster

Figure 3: Workload-specific EDA clusters

In Figure 3, you can see that the only centralized resources are those required for every type of job. This includes the job scheduler, license server, and directory or authentication services. All of the compute resources and NFS servers are dedicated to specific projects so that they can be sized according to that project's needs.

Job Scheduler Integration

Job schedulers such as IBM Platform LSF, Adaptive PBS Pro, and Univa Grid Engine (or their open source alternatives) are typically used in the EDA industry to manage compute resources, optimize license usage, and coordinate jobs. When you migrate to AWS, you might choose to retain these existing schedulers in order to minimize the impact on your end-user workflows and processes. Most of these job schedulers have some form of native integration with AWS where you can use the head node to automatically launch cloud resources when there are jobs pending in the queue. If you want to run these schedulers on AWS, refer to the documentation of your specific job management tool in order to understand how to configure it to manage cloud resources.

Data Access and Transfer

When you first consider running workloads in the cloud, you might envision a bursting scenario where cloud resources are set up as an augmentation to your existing on-premises compute cluster. While you can use this model successfully, data movement presents a significant challenge when building an architecture to support bursting in a seamless way. Your organization might see the most success if you consider bursting on a project-by-project basis and choose to run entire workflows on AWS, thereby freeing up existing resources to handle other tasks. By approaching cloud resources this way, you can use much simpler data transfer mechanisms because you aren't trying to keep two independent data centers in sync.

Licensing

Application licensing is required for most EDA workloads, both on-premises and on AWS. From a technical standpoint, managing and accessing licenses is unchanged when migrating to AWS.

License Server Access

On AWS, each Amazon EC2 instance launched is provided with a unique hostname and hardware (MAC) address using Amazon elastic network interfaces that cannot be cloned or spoofed. Therefore, traditional license server technologies (for example, Flexera) work natively on AWS without any modification. The inability to clone license servers, which is prevented by AWS by not allowing the duplication of MAC addresses, also provides EDA software vendors with increased confidence that EDA software can be deployed and used in a secure manner.

Because of the connectivity options available, which include the use of VPNs and AWS Direct Connect, you can run your license servers on AWS using an Amazon EC2 instance as a license server or within your own data centers. By allowing connectivity through a VPN or AWS Direct Connect between cloud resources and on-premises license servers, AWS enables users to seamlessly run workloads in any location without having to split licenses and dedicate them to specific groups of compute resources.

Working with EDA Vendors

AWS works closely with thousands of independent software vendors (ISVs) that deliver solutions to customers on AWS using methods that may include SaaS, PaaS, customer self-managed, and BYOL models. In the semiconductor sector, AWS works closely with major vendors of EDA software. AWS can assist ISVs and your organization with deployment best practices as described in this whitepaper.

EDA vendors that are members of the AWS Partner Network (APN) have access to a variety of tools, training, and support that are provided directly to the EDA vendor, which benefits EDA end-customers. These Partner Programs are designed to support the unique technical and business requirements of APN members by providing them with increased support from AWS, including access to AWS partner team members who specialize in design and engineering applications. In addition, AWS has a growing number of Consulting Partners who can assist EDA vendors and their customers with EDA cloud migration.

Security and Governance in the AWS Cloud

The cloud offers a wide array of tools and configurations that enable your organization to protect your data and IP in ways that are difficult to do with traditional on-premises environments. This section outlines some of the ways you can do this in the AWS Cloud.

Isolated Environments for Data Protection and Sovereignty

In the following reference architecture, each service provided is built for both high availability and security with strict rules that govern how data can be transferred throughout the environment. Security groups are similar to firewalls—they ensure that access to specific resources is tightly controlled. Subnets containing compute and storage resources can be isolated so that they don’t have any direct access to the internet. Users who need to access the environment must first connect to the Bastian Node (also commonly referred to as a jump box) through secure protocols, like SSH. From there, they can log into interactive desktops or job schedulers as permitted through your organization's security policies.

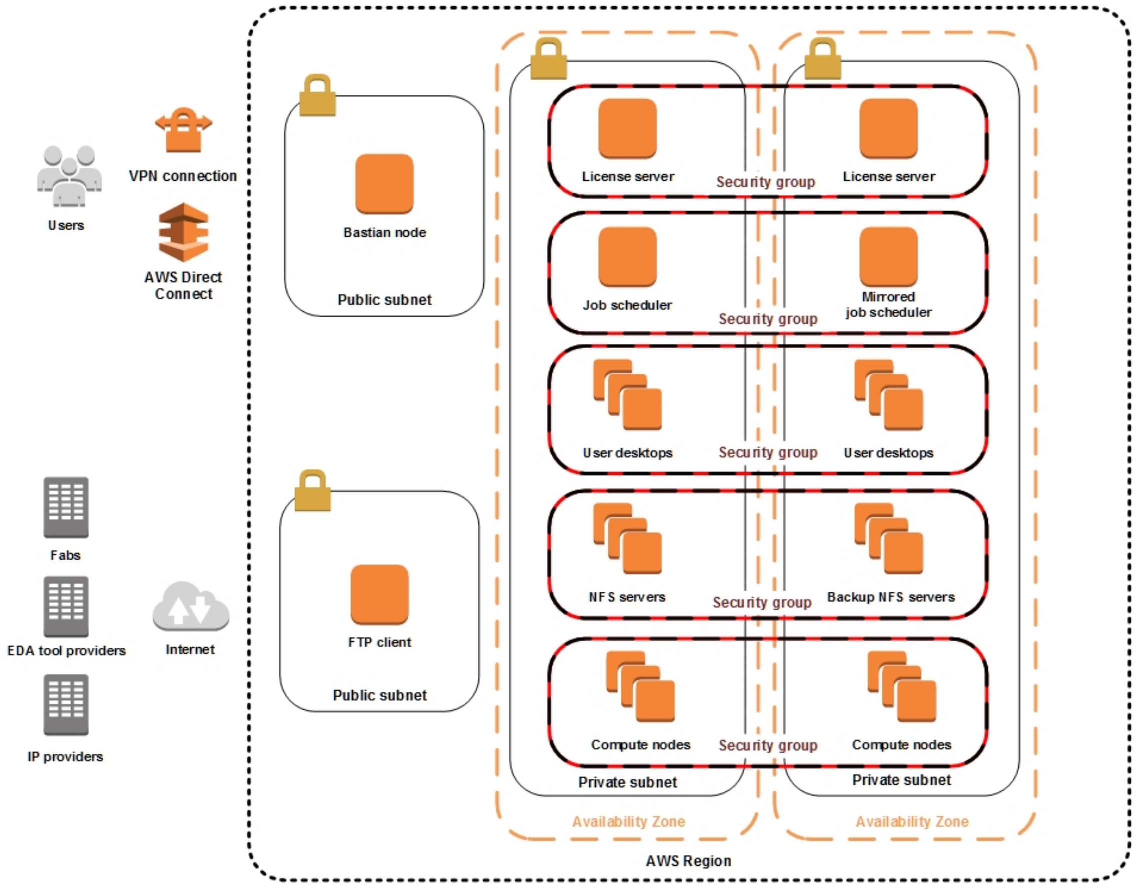

Figure 4: Highly available and secure EDA architecture

Often times secure FTP is required in environments such as this. Organization use secure FTP to download tools from vendors, copy completed designs to fabrication facilities, or to update IP from suppliers. In order to do this securely, you can set up an FTP client in an isolated subnet that has limited access to external IP addresses as necessary. You segment this client from the rest of the network, and configure strict controls and monitoring to ensure that everything on that server is secure.

User Authentication

When it comes to managing users and access to compute nodes, you can adapt the technologies that you use today to work in the same way on AWS. Many organizations already have existing LDAP, Microsoft Active Directory, or NIS services that they use for authentication. Almost all of these services provide replication and functionality to support multiple data centers. With the appropriate network and VPN setup in place, you can manage these systems on AWS using the same methods and configurations as you would for any remote data center configuration.

If your organization is interested in running an isolated directory on the cloud, you have a number of options to choose from. If you want to use a managed solution, AWS Directory Service for Microsoft Active Directory (Standard) is a popular choice.[8] AWS Microsoft AD (Standard Edition) is a managed Microsoft Active Directory (AD) that is optimized for small and midsize businesses (SMBs). Other options include running your own LDAP or NIS infrastructure on AWS, and more current solutions like FreeIPA.

Network

AWS employs a number of technologies that allow you to isolate components from each other and control access to the network.

Amazon VPC

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications.

You can easily customize the network configuration for your Amazon VPC. For example, you can create a public-facing subnet for your FTP and Bastian servers that has access to the internet. Then, you can place your design and engineering systems in a private-facing subnet with no internet access. You can leverage multiple layers of security, including security groups and network access control lists, to help control access to EC2 instances in each subnet.

Additionally, you can create a hardware virtual private network (VPN) connection between your corporate data center and your VPC and leverage the AWS Cloud as an extension of your organization's data center.

Security Groups

Amazon VPC provides advanced security features such as security groups and network access control lists (ACLs) to enable inbound and outbound filtering at the instance level and subnet level. A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign the instance to up to five security groups.

Amazon VPC provides advanced security features such as security groups and network access control lists to enable inbound and outbound filtering at the instance level and subnet level. Security groups control inbound and outbound traffic for your instances. Network ACLs control inbound and outbound traffic for your subnets. In most cases, security groups can meet your needs. However, you can also use network ACLs if you want an additional layer of security for your VPC. For more information, see Security in the *Amazon VPC User Guide*.[9]

You can create a flow log on your Amazon VPC or subnet to capture the traffic that flows to and from the network interfaces in your VPC or subnet. You can also create a flow log on an individual network interface. Flow logs are published to Amazon CloudWatch Logs.

Data Storage and Transfer

AWS offers many ways to protect data, both in flight and at rest. Many third-party storage vendors also offer additional encryption and security technologies in their own implementations of storage in the AWS Cloud.

AWS Key Management Service (KMS)

AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data, and uses Hardware Security Modules (HSMs) to protect the security of your keys. AWS KMS is integrated with other AWS services including Amazon EBS, Amazon S3, Amazon Redshift, Amazon Elastic Transcoder, Amazon WorkMail, Amazon Relational Database Service (Amazon RDS), and others, to help you protect the data you store with these services. AWS KMS is also integrated with AWS CloudTrail to provide you with logs of all key usage to help meet your regulatory and compliance needs.

With AWS KMS, you can create master keys that can never be exported from the service. You use the master keys to encrypt and decrypt data based on policies that you define.

Amazon EBS Encryption

Amazon Elastic Block Store (Amazon EBS) encryption offers you a simple encryption solution for your EBS volumes requiring you to build, maintain, and secure your own key management infrastructure. When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:

- Data at rest inside the volume
- All data moving between the volume and the instance
- All snapshots created from the volume

The encryption occurs on the servers that host EC2 instances, providing encryption of data-in-transit from EC2 instances to Amazon EBS storage.

Amazon S3 Encryption

When you use encryption with Amazon S3, Amazon S3 encrypts your data at the object level as Amazon S3 writes the data to disks in AWS data centers and decrypts your data when you access it. As long as you authenticate your request and you have access permissions, there is no difference in how you access encrypted or unencrypted objects.

AWS KMS uses customer master keys (CMKs) to encrypt your Amazon S3 objects. You use AWS KMS via the **Encryption Keys** section in the AWS Identity and Access management (AWS IAM) console or via AWS KMS APIs to create encryption keys, define the policies that control how keys can be used, and audit key usage to ensure that they are used correctly. You can use these keys to protect your data in Amazon S3 buckets.

Server-side encryption with AWS KMS-managed keys (SSE-KMS) provides the following:

- You can choose to create and manage encryption keys yourself, or you can choose to generate a unique default service key on a customer/service/region level.

- The ETag in the response is not the MD5 of the object data.
- The data keys used to encrypt your data are also encrypted and stored alongside the data they protect.
- You can create, rotate, and disable auditable master keys in the IAM console.
- The security controls in AWS KMS can help you meet encryption-related compliance requirements.

If you require server-side encryption for all objects that are stored in your bucket, Amazon S3 supports bucket policies that you can use.

Because access to Amazon S3 is provided over HTTP endpoints, you can set up policies on a bucket to ensure that all data transfer in and out occurs over a TLS connection to ensure that data is also encrypted in transit.

Governance and Monitoring

AWS provides several services that you can use to enforce governance and monitor your AWS Cloud deployment:

- **AWS Identity and Access Management (IAM)** – Enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources. For more information, see the AWS IAM User Guide.[10]
- **Amazon CloudWatch** – Enables you to monitor your AWS resources in near real-time, including EC2 instances, EBS volumes, and S3 buckets. Metrics such as CPU utilization, latency, and request counts are provided automatically for these AWS resources. You can also provide CloudWatch access to your own logs or custom application and system metrics, such as memory usage, transaction volumes, or error rates, and CloudWatch can monitor these too. For more information, see the Amazon CloudWatch User Guide.[11]
- **Amazon CloudWatch Logs** – Use to monitor, store, and access your log files from EC2 instances, AWS CloudTrail, and other sources. You can then retrieve the associated log data from CloudWatch Logs. You can create alarms in CloudWatch and receive notifications of particular

API activity as captured by CloudTrail and use the notification to perform troubleshooting. For more information, see the Amazon CloudWatch Log User Guide.[12]

- **AWS CloudTrail** – Enables you to log, continuously monitor, and retain events related to API calls across your AWS infrastructure. CloudTrail provides a history of AWS API calls for your account, including API calls made through the AWS Management Console, AWS SDKs, command line tools, and other AWS services. For more information, see the AWS CloudTrail User Guide.[13]
- **AWS Config** – Use to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations. For more information, see the AWS Config Developer Guide.[14]
- **AWS Organizations** – Offers policy-based management for multiple AWS accounts. With Organizations, you can create Service Control Policies (SCPs) that centrally control AWS service use across multiple AWS accounts. Organizations helps simplify the billing for multiple accounts by enabling you to setup a single payment method for all the accounts in your organization through consolidated billing. You can ensure that entities in your accounts can use only the services that meet your corporate security and compliance policy requirements. For more information, see the AWS Organizations User Guide.[15]

Contributors

The following individuals and organizations contributed to this document:

- Adam Boeglin, Senior HPC Solutions Architect, Amazon Web Services
- Mark Duffield, Principal HPC Solutions Architect, Amazon Web Services
- David Pellerin, Principal Business Development for Infotech/Semiconductor, Amazon Web Services
- Nafea Bshara, VP/Distinguished Engineer, Amazon Web Services

Document Revisions

Date	Description
October 2017	First publication

Notes

[1] https://aws.amazon.com/about-aws/global-infrastructure/?hp=tile

[2] https://aws.amazon.com/ec2/virtualcores/

[3] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSOptimized.html#ebs-optimization-support

[4] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html

[5] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonLinuxAMIBasics.html

[6] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/processor_state_control.html

[7] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html

[8] http://docs.aws.amazon.com/directoryservice/latest/admin-guide/directory_simple_ad.html

[9] http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Security.html

[10] http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

[11] http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

12 http://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html

13 http://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html

14 http://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html

15 http://docs.aws.amazon.com/organizations/latest/userguide/orgs_introduction.html