Optimizing Electronic Design Automation (EDA) Workflows on AWS

September 2018

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

Semiconductor and electronics companies using electronic design automation (EDA) can significantly accelerate their product development lifecycle and time to market by taking advantage of the near infinite compute, storage, and resources available on AWS. This whitepaper presents an overview of the EDA workflow, recommendations for moving EDA tools to AWS, and the specific AWS architectural components to optimize EDA workloads on AWS.

Introduction

The workflows, applications, and methods used for the design and verification of semiconductors, integrated circuits (ICs), and printed circuit boards (PCBs) 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 IT teams that provide infrastructure support for semiconductor companies—must invest in increasingly large server farms and high-performance storage systems to enable higher quality and faster turnaround of semiconductor test and validation. The introduction of new and upgraded IC fabrication technologies may require large amounts of compute and storage, for relatively short times, to enable rapid completion of hardware regression testing or to recharacterize design IP.

Semiconductor companies today use Amazon Web Services (AWS) to take advantage of a more rapid, flexible deployment of CAE and EDA infrastructure, from the complete IC design workflow, from register-transfer-level (RTL) design, to the delivery of GDSII files to a foundry for chip fabrication. AWS compute, storage, and higher level services are available on a dynamic, as-needed basis without the significant upfront capital expenditure that is typically required for performance-critical EDA workloads.

EDA Overview

EDA workloads comprise workflows and a supporting set of software 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 as EDA software, which expedites and reduces time to silicon tape-out and fabrication. EDA is a highly iterative engineering process that can take from months, and in some cases 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 (provided by IP developers as generic gate-level netlists) are either designed in-house by a semiconductor company, or purchased from a third-party IP vender. IP cores themselves requires EDA workflows for design and verification, and to characterize performance for specific IC fabrication technologies. These IP cores are used in combination with IC-specific, custom-designed components, to create a complete IC that often includes a complex system-on-chip (SoC) making use of one of more embedded CPUs, standard peripherals, I/O, and custom analog and/or digital components.

The complete IC itself, with all its IP cores and custom components, then requires large amounts of EDA processing for full-chip verification—including modeling (that is, simulating) all of the components within the chip. This modeling, which includes HDL source-level validation, physical synthesis, and initial verification (for example, using techniques such as formal verification), is known as the *front-end design*. The physical implementation, which includes floor planning, place and route, timing analysis, design-rule-check (DRC), 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, for historical reasons, as *tapeout*. When completed, the file is sent to a fabrication facility (a *foundry*), which may or may not be operated by the semiconductor company, where a silicon wafer is manufactured. This wafer, containing perhaps thousands of individual ICs, is then inspected, cut into dies that are themselves tested, packaged into chips that are tested again, and assembled onto a board or other system through highly automated manufacturing processes.

All of these steps in the semiconductor and electronics supply chain can benefit from the scalability of cloud.

Benefits of the AWS Cloud

Before discussing the specifics of moving EDA workloads to AWS, it is 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 and time to market. Your organization can achieve this by scaling out your compute needs to meet the demands of the jobs waiting to be processed. AWS uses per

second billing for our compute resources, allowing you to optimize cost by only paying for what you use, down to the second. By scaling horizontally, you can run more compute servers (that is, Amazon Elastic Compute Cloud [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 design regression 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 comprise regions and Availability Zones (AZs). 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 design achieves the greatest possible fault tolerance and stability. Resources are not 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. Additionally, this large global footprint enables you to position computing resources near your IC design engineers in situations where low-latency performance is important. For more information, refer to AWS Global Infrastructure.

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, launch entire clusters of servers, and scale these clusters up and down, uniquely optimizing each cluster for specific applications, and for 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 might need to have a cluster of 100 machines (representing over 2,000 CPU cores) with a specific memory-to-core ratio and a specific storage configuration. With AWS, you can deploy and run this cluster, dedicated only for this task, for only 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 AWS to augment or replace these geographically separated IT resources, you can pool all of 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.

Paths for Migrating EDA Workflows to AWS

When you begin the migration of EDA workflows to AWS, you will find there are many parallels with managing traditional EDA deployments across multiple data centers. Larger organizations in the semiconductor industry typically have multiple data centers that are geographically segregated because of the distributed nature of their 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, you 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.

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. Although 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 benefit if you consider bursting on a project-by-project basis and choose to run entire workflows on AWS, thereby freeing up existing on-premises resources to handle other tasks. By approaching cloud resources this way, you can use much simpler data transfer mechanisms because you are not trying to sync data between AWS and your data centers.

Consider what Data to Move to Amazon S3

Prior to moving your EDA tools to AWS, consider the processes and methods that will be in place as you move from initial experiments to full production. For example,

consider what data will be needed for an initial performance test, or for a first workflow proof of concept (POC).

Data is gravity, and moving only the limited amount of data needed to run your EDA tools to an Amazon Simple Storage Service (Amazon S3) bucket allows for flexibly and agility when building and iterating your architecture on AWS. There are several benefits to storing data in Amazon S3; for an EDA POC, using Amazon S3 allows you to iterate quickly, as the S3 transfer speed to an EC2 instance is up to 25 Gbps. With your data stored in an S3 bucket, you can more quickly experiment with different EC2 instance types, and also experiment with different working-storage options, such as creating and tuning temporary shared file systems.

Deciding what data to transfer is dependent on the tools or designs you are planning to use for the POC. We encourage customers to start with a relatively small amount of POC data; for example, only the data required to run a single simulation job. Doing so allows you to quickly gain experience with AWS and build an understanding of how to build production-ready architecture on AWS while in the process of running an initial EDA POC workload.

Dependencies

Semiconductor design environments often have complex dependencies that can hinder the process of moving workflows to AWS. We can work with you to build an initial proof of concept or even a complex architecture. However, it is the designer's or tool engineer's responsibility to unwind any legacy on-premises data dependencies. The initial POC process requires effort to determine which dependencies, such as shared libraries, need to be moved along with project data. There are tools available that help you build a list of dependencies, and some of these tools yield a file manifest that expedites the process of moving data to AWS. For example, one tool is Ellexus Container Checker, which can be found on the AWS Marketplace.

Dependencies can include authentication methods (for example, NIS), shared file systems, cross organization collaboration, and globally distributed designs. (Identifying and managing such dependencies is not unique to cloud migration; semiconductor design teams face similar challenges in any distributed EDA environment.)

Another approach may be to launch a net-new semiconductor project on AWS, which should significantly reduce the number of legacy dependencies.

Suggested EDA Tools for Initial Proof of Concept (POC)

An HDL compile and simulation workflow may be the fastest approach to launching an EDA POC on AWS, or creating a production EDA environment. HDL files are typically not large, and the ability to use an on-premises license server (via VPN), reduces the additional effort of moving your licensing environment to AWS. HDL compile and simulation workflows are representative of other EDA workloads, including their need for shared file systems and some form of job scheduling.

Cloud-Optimized Traditional Architecture

On AWS, compute and storage resources are available on-demand, allowing you to launch on what you need and when you need it. This enables a different approach to architecting your semiconductor design environment. Rather than having one large cluster where multiple projects are running, you can use AWS to launch multiple clusters. Because you can configure compute resources to increase or decrease 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-size compute and storage, and environment isolation.

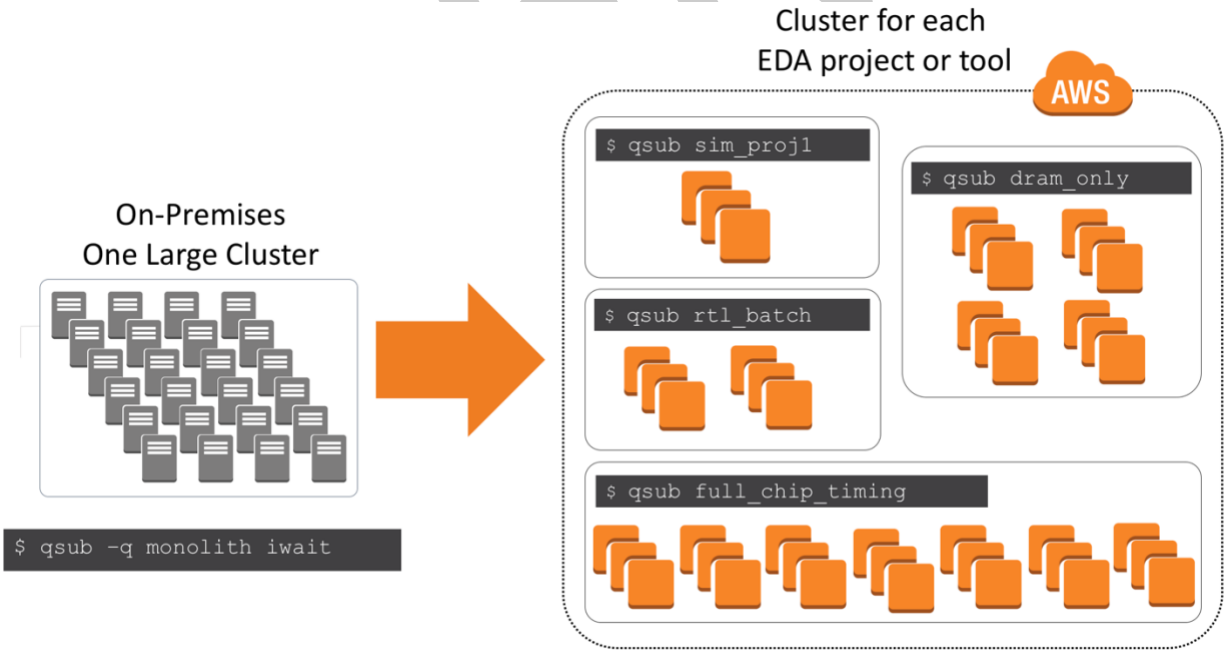


Figure 1: Workload-specific EDA clusters on AWS

As seen in Figure 1, moving to AWS allows you to launch a separate set of resources for each of your EDA workloads (for example, a cluster). This multi-cluster approach can also be used for global and cross-organizational collaboration. The multi-cluster approach can be used, for example, to dedicate and manage specific cloud resources for specific projects, encouraging organizations to use only the resources required for their project.

Job Scheduler Integration

The EDA workflow that you build on AWS can be a similar environment to the one you have in your on-premises data center. Many, if not all, of the same EDA tools and applications running in your data center, as well as orchestration software, can also be run on AWS. 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 and prioritize jobs. When you migrate to AWS, you may choose to use these existing schedulers essentially unchanged, to minimize the impact on your end-user workflows and processes. Most of these job schedulers already have some form of native integration with AWS, allowing you to use the master node to automatically launch cloud resources when there are jobs pending in the queue. You should refer to the documentation of your specific job management tool for the steps to automate resource allocation and management on AWS.

Building an EDA Architecture on AWS

Building out your production-ready EDA workflow on AWS requires an end-to-end examination of your current environment. This examination begins with the operating system you are using for running your EDA tools, as well as your job scheduling and user management environments. AWS allows for a mix of architectures when moving semiconductor design workloads, and you can leverage some combination of the following two approaches:

- Build an architecture similar to a traditional cluster, using traditional job scheduling software, but ensuring that a cloud native approach is used.
- Use more cloud-native methods, such as AWS Batch, which uses containerization of your applications.

Where needed, we will make the distinction when using AWS Batch can be advantageous, for example when running massively parallel parameter sweeps.

Hypervisors: Nitro and Xen

Amazon EC2 instances use a hypervisor to divide resources on the server, so that each customer has separate CPU, memory, and storage resources for just that customer's instance. We do not use the hypervisor to share resources between instances, except for the T* family. On previous generation instance types, for example the C4 and R4 families, EC2 instances are virtualized using the Xen hypervisor. In current-generation instances, for example C5, R5, and Z1d, we are using a specialized piece of hardware and a highly customized hypervisor based on KVM. This new hypervisor system is called Nitro. At the time of this writing, these are the Nitro based instances: Z1d, C5, C5d, M5, M5d, R5, R5d.

Launching Nitro based instances requires that specific drivers for networking and storage be installed and enabled before the instance can be launched. We provide the details for this configuration in the next section.

AMI and Operating System

AWS has built-in support for numerous operating systems (OSs). For EDA users, CentOS, Red Hat Enterprise Linux, and Amazon Linux 2 are used more than other operating systems. The operating system and the customizations that you have made in your on-premises environment are the baseline for building out your EDA architecture on AWS. Before you can launch an EC2 instance, you must decide which Amazon Machine Image (AMI) to use. An AMI contains the OS, any required OS and driver customizations, and may also include the application software. For EDA, one approach is to launch an instance from an existing AMI, customize the instance after launch, and then save this updated configuration as a custom AMI. Instances launched from this new custom AMI include the customizations that you made when you created the AMI.

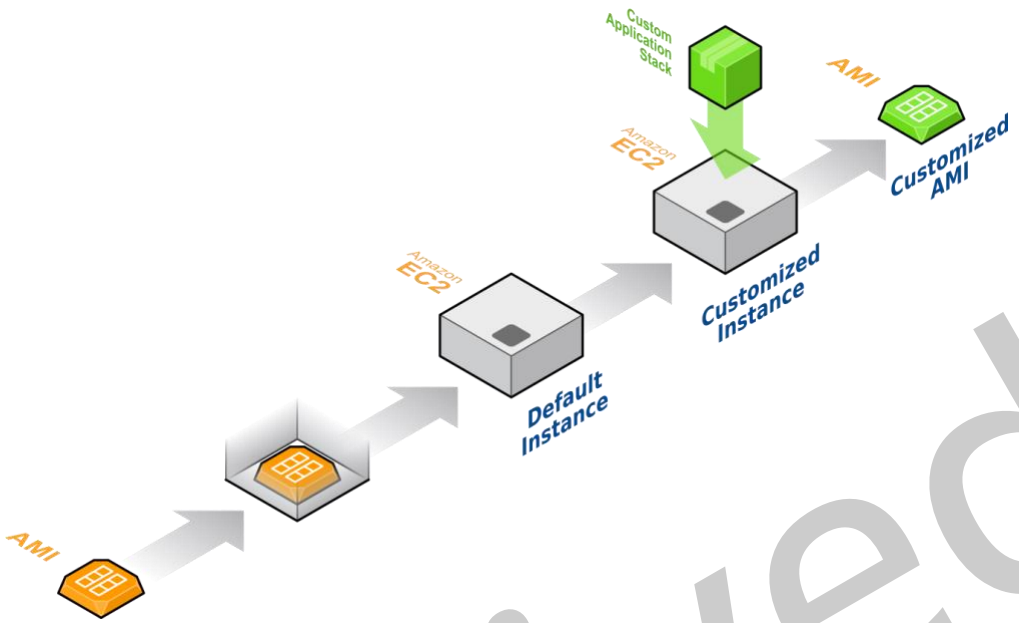


Figure 2: Use Amazon provided AMI to build a Customized AMI

Figure 2 illustrates the process of launching an instance with an AMI. You can select the AMI from the AWS Console or from the AWS Marketplace, and then customize that instance with your EDA tools and environment. After that, you can use the customized instance to create a new, customized AMI that you can then use to launch your entire EDA environment on AWS. Note also that the customized AMI that you create using this process can be further customized. For example, you can customize the AMI to add additional application software, load additional libraries, or apply patches, each time the customized AMI is launched onto an EC2 instance.

As of this writing, we recommend these OS levels for EDA tools (more detail on OS versions is provided in following sections):

- Amazon Linux and Amazon Linux 2 (verify certification with EDA tool vendors)
- CentOS 7.4 or 7.5
- Red Hat Enterprise Linux 7.4 or 7.5

These OS levels have the necessary drivers already included to support the current instance types, which include Nitro based instances. If you are not using one of these levels, you must perform extra steps to take advantage of the features of our current instances. Specifically, you must build and enable enhanced networking, which relies on

the elastic network adaptor (ENA) drivers. See *Network* and *Optimizing EDA Tools on AWS* for more detailed information on ENA drivers and AMI drivers.

If you use an instance with Nitro (Z1d, C5, C5d, M5, M5d, R5, R5d), you must use an AMI that has the AWS ENA driver built and enabled, and the NVMe drivers installed. At this time, a Nitro based instance does not launch unless you have these drivers. These OS levels include the required drivers:

- CentOS 7.4 or later
- Red Hat Enterprise Linux 7.4 or later
- Amazon Linux or Amazon Linux 2 (current versions)

To verify that you can launch your AMI on a Nitro based instance, first launch the AMI on a Xen based instance type, and then run the `c5_m5_checks_script.sh` script found on the awslabs GitHub repo at awslabs/aws-support-tools/EC2/C5M5InstanceChecks/

The script analyzes your AMI and determines if it can run on a Nitro based instance. If it cannot, it displays recommended changes.

You can also import your own on-premises image to use for your AMI. This process includes extra steps, but may result in time savings. Before importing an on-premises OS image, you first require a VM image for your OS. AWS supports certain VM formats (for example, Linux VMs that use VMware ESX) that must be uploaded to an S3 bucket, and subsequently converted into an AMI. Detailed information and instructions can be found at https://aws.amazon.com/ec2/vm-import/

The same operating system requirements mentioned above are also applicable to imported images (that is, you should use CentOS/RHEL 7.4 or 7.5, Amazon Linux or Amazon Linux 2).

Compute

Although 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. When running EDA software on AWS, you should choose 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.

The compute-optimized instance family features instances that have the highest clock frequencies available on AWS, and typically enough memory to run some memory intensive workloads.

Typical EDA use cases for compute-optimized instance types:

- Simulations
- Synthesis
- Formal verification
- Regression tests

Z1d for EDA Tools

AWS has recently announced a powerful new instance type that is well optimized for EDA applications. The faster clock speed on the Z1d instance, with up to 4 GHz sustained Turbo performance, allows for EDA license optimization while achieving faster-time to results. The Z1d uses an AWS specific Intel Xeon Platinum 8000-series (Skylake) processor and is the fastest AWS instance type. The following list summarizes the features of the Z1d instance:

- Sustained all core frequency of up to 4.0 GHz
- Six different instance sizes with up to 24 cores (48 threads) per instance
- Total memory of 384 GiB
- Memory to core ratio of 16 GiB RAM per core
- Includes local Instance Store NVMe storage (as much as 1.8 TiB)
- Optimized for EDA and other high performance workloads

Additional Compute-Optimized Instances

C5, C5d, C4

In addition to the Z1d, the C5 instance features up to 36 cores (72 threads) and up to 144 GiB of RAM. The processor used in the C5 is the same as the Z1d, the Intel Xeon Platinum 8000-series (Skylake), but also includes a base clock speed of 3.0 GHz and the ability to turbo boost up to 3.5 GHz.

The C5d instance is the same configuration as the C5, but offers as much as 1.8 TiB of local NVMe SSD storage.

Previous generation C4 instances are also commonly used by EDA customers and still remain a suitable option for certain workloads, such as those that are not memory-intensive.

Memory Optimized Instances

Z1d, R5, R5d, R4, X1, X1e

The Z1d instance is not only compute-optimized, but memory optimized as well, including 384 GiB of total memory. The Z1d has the highest clock frequency of any instance, and with the exception of our X1 and X1e instances, is equal to the most memory per core (16 GiB/core). If you require larger amounts of memory than what is available on the Z1d, consider another memory-optimized instance, such as the R5, R5d, R4, X1, or X1e.

Typical EDA use cases for memory-optimized instance types:

- Place and route
- Static timing analysis
- Physical verification
- Batch mode RTL simulation (multithread optimized tools)

The R5 and R5d have the same processor as the Z1d and C5, the Intel Xeon Platinum 8000-series (Skylake). With the largest R5 and R5d instance types having up to 768 GiB memory, EDA workloads that could previously only run on the X1 or X1e, can now run on the R5 and R5d instances. These recently released instances are serving as a drop-in replacement for the R4 instance, for both place and route, as well as batch mode RTL simulation. The R4.16xlarge instance is viable option, with a high core count (32) and 15 GiB/core ratio. For this reason, we see a large number of customers using the R4.16xlarge instance type.

The X1 and X1e instance types can also be used for memory intensive workloads; however, testing of EDA tools by Amazon internal silicon teams has indicated that most EDA tools will run well on the Z1d, R4, R5, or R5d instances. The need for the amount of memory provided on the X1 (1952 GiB) and X1d (3,904 GiB) has been relatively infrequent for semiconductor design.

Hyper-Threading

Amazon EC2 instances support Intel Hyper-Threading Technology (HT Technology), which enables multiple threads to run concurrently on a single Intel Xeon CPU core.

Each thread is represented as a virtual CPU (vCPU) on the instance. An instance has a default number of CPU cores, which varies according to instance type. Each vCPU is a hyperthread of an Intel Xeon CPU core, except for T2 instances. You can specify the following CPU options to optimize your instance for semiconductor design workloads:

- **Number of CPU cores**: You can customize the number of CPU cores for the instance. This customization may optimize the licensing costs of your software with an instance that has sufficient amounts of RAM for memory-intensive workloads but fewer CPU cores.
- **Threads per core**: You can disable Intel Hyper-Threading Technology by specifying a single thread per CPU core. This scenario applies to certain workloads, such as high performance computing (HPC) workloads.

You can specify these CPU options during instance launch (currently on support through the AWS Command Line Interface [AWS CLI], an AWS software development kit [SDK], or the Amazon EC2 API only). There is no additional or reduced charge for specifying CPU options. You are charged the same amount as instances that are launched with default CPU options. Refer to Optimizing CPU Options in the *Amazon Elastic Compute Cloud User Guide for Linux Instances* for more details and rules for specifying CPU options.

Divide the vCPU number by 2 to find the number of physical cores on the instance. You can disable HT Technology if you determine that it has a negative impact on your application's performance. See *Optimizing EDA Tools on AWS* for details on disabling Hyper-Threading.

Table 1 lists the instance types that are typically used for EDA tools.

Table 1: Instance specifications suitable for EDA workloads

Instance Name	*Max Core Count	CPU Clock Frequency	Max Total RAM in GiB	Memory to core ratio, GiB / core	Local NVMe
Z1d	24	4.0 GHz	384	16	Yes
R5 / R5d	48	Up to 3.1 GHz	768	16	Yes on R5d
R4	32	2.3 GHz	488	15.25	
M5 / M5d	48	Up to 3.1 GHz	384	8	Yes on M5d
C5 / C5d	36	Up to 3.5 GHz	144	4	Yes on C5d

Instance Name	*Max Core Count	CPU Clock Frequency	Max Total RAM in GiB	Memory to core ratio, GiB / core	Local NVMe
X1	64	2.3 GHz	1,952	30.5	Yes
X1e	64	2.3 GHz	3,904	61	Yes
C4	18	2.9 GHz boost to 3.5	60	3.33	

*NOTE: AWS uses vCPU (which is an Intel Hyper-Thread) to denote processors, for this table we are using cores.

Network

Amazon enhanced networking technology enables instances to communicate at up to 25 Gbps for current-generation instances and up to 10 Gbps for previous-generation instances. In addition, enhanced networking reduces latency and network jitter.

Enhanced networking is enabled by default on these operating system levels:

- Amazon Linux
- Amazon Linux 2
- CentOS 7.4 and 7.5
- Red Hat Enterprise Linux 7.4 and 7.5

If you have an older version of CentOS or RHEL you can enable enhanced networking by installing the network module and updating the enhanced network adapter (ENA) support attribute for the instance. For more information about enhanced networking, including build and install instructions, refer to the Enhanced Networking on Linux page in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.

Storage

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 network file systems (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 increased job run times 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, resulting in reduced run times for large batch workloads. Achieving these benefits may require some amount of EDA workflow rearchitecting, but the benefits of making these optimizations can be numerous.

Types of Storage on AWS

Before discussing the different options for deploying EDA storage, it is 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 a 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 required to run semiconductor workloads.

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).

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 is possible to achieve up to 80,000 IOPS and 1,750 MB/s from a single Amazon EC2 instance.

It is important to choose the correct EBS volume types when building your EDA architecture on AWS. Table 2 shows the EBS volumes types that you should consider.

Table 2: EBS Volume Types

	io1	gp2*	st1	sc1
Volume Type	Provisioned IOPS SSD	General Purpose SSD	Throughput Optimized HDD	Cold HDD
Volume Size	4 GB - 16 TB	1 GB - 16 TB	500 GB - 16 TB	500 GB - 16 TB
Max IOPS/Volume**	32,000	10,000	500	250
Max Throughput/Volume	500 MB/s	160 MB/s	500 MB/s	250 MB/s

*Default volume type

**io1/gp2 based on 16K I/O size, st1/sc1 based on 1 MB I/O size

When choosing your EBS volume types, consider the performance characteristics of each EBS volume. This is particularly important when building a NFS server or another file system solutions. Achieving the maximum capable performance of an EBS volume depends on the size of the volume. Additionally, the gp2, st1, and sc1 volume types use a burst credit system, and this should be taken in to consideration as well.

Each AWS EC2 instance type has a throughput and IOPS limit. For example, the Z1d.12xlarge has EBS limits of 1.75 GB/s and 80,000 IOPS. (For a chart that shows the Amazon EBS throughput expected for each instance type, refer to Instance Types that Support EBS Optimization in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.) To achieve these speeds, you must stripe multiple EBS volumes together as each volume has its own throughput and IPOS limit. Refer to Amazon EBS Volume Types in the *Amazon Elastic Compute Cloud User Guide for Linux Instances* for detailed information about throughput, IOPS, and burst credits.

Enhancing Scalability with Dynamic EBS Volumes

Semiconductor design has a long history of over-provisioning hardware to meet the demands of backend workloads that may not be run for months or years after the customer specifications are received. On AWS, you provision only the resources you need, when you need them. 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.

A key feature of EBS storage is elastic volumes (available on all current generation EBS volumes attached to current-generation EC2 instances). This feature allows you to provision a volume that meets your application requirements today, and as your requirements change, allows you to increase the volume size, adjust performance, or change the volume type while the volume is in use. You can continue to use your application while the change takes effect.

An on-premises installation normally requires manual intervention to adjust storage configurations. Leveraging EBS elastic volumes and other AWS services, you can automate the process of resizing your EBS volumes. Figure 3 shows the automated process of increasing the volume size using Amazon CloudWatch (metrics and monitoring service and AWS Lambda (an event-driven, serverless compute service). The volume increase event is triggered (e.g. usage threshold) using a CloudWatch alarm and a Lamba function. The resulting increase is automatically detected by the operating system, and a subsequent file system grow operation resizes the file system.

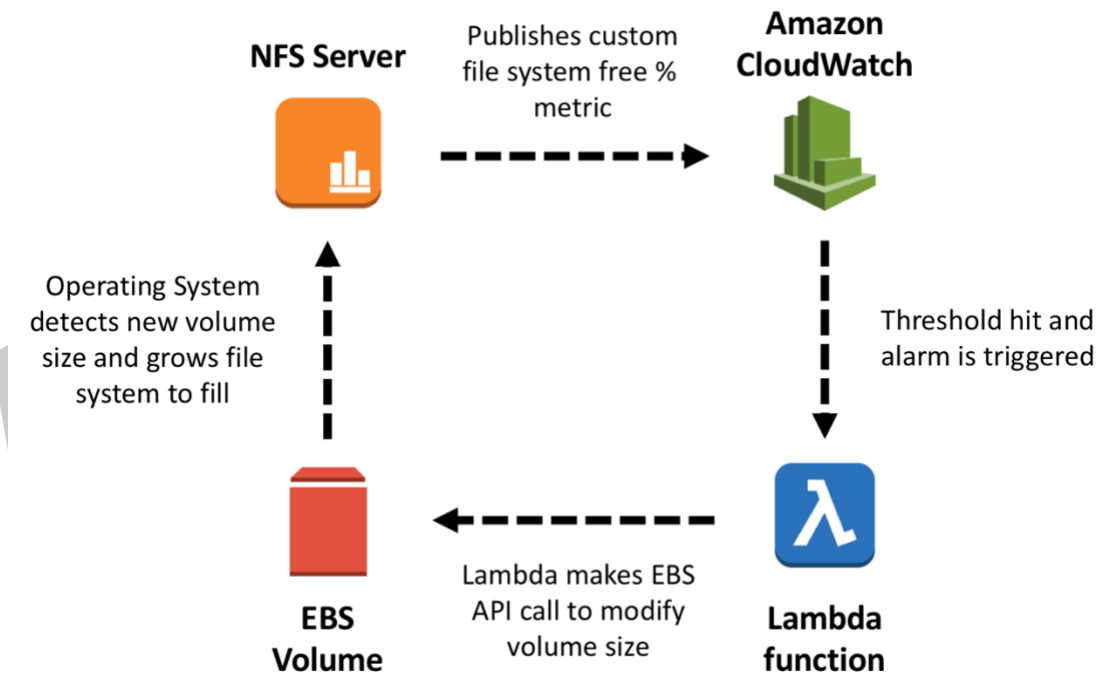


Figure 3: Lifecycle for automatically resizing an EBS volume

Instance Storage

For use cases where the performance of Amazon EBS is not sufficient on a single instance, Amazon EC2 instances with Instance Store are available. Instance Store is block-level storage that is physically attached to the instance. As the storage is directly attached to the instance, it can provide significantly higher throughput and IOPS than is

available through network based storage, similar to Amazon EBS. However, because the storage is locally attached to the instance, data on the Instance Store does not persist when you stop or terminate the instance. Additionally, hardware failures on the instance would likely result in data loss. For these reasons, instance Store is recommended for temporary scratch space or for data replicated off of the instance (for example, Amazon S3). You can increase durability by choosing an instance with multiple NVMe devices, and create a RAID set with one or more parity devices.

The I3 instance family and the recently announced Z1d, C5d, M5d and R5d instances are well-suited for EDA workloads requiring a significant amount of fast local storage, such as scratch data. These instances use NVMe based storage devices and are designed for the highest possible IOPS. The Z1d and C5d instances each have up to 1.8 TiB of local instance store, and the R5d and M5d instances each have up to 3.6 TiB of local instance store.

The i3.16xlarge can deliver 3.3 million random IOPS at 4 KB block size and up to 16 GB/s of sequential disk throughput. This performance makes the i3.16xlarge well-suited for serving file systems for scratch or temporary data over NFS.

Table 3 shows the instance types typically found in the semiconductor space that have instance store.

Table 3: Instances typically found in the EDA space with Instance Store

Instance Name	Max Raw Size TiB	Number and size of NVMe SSD (GiB)
I3	15.2 TiB	8 x 1920
Z1d	1.8 TiB	2 x 900
R5d	3.6 TiB	4 x 900
M5d	3.6 TiB	4 x 900
C5d	1.8 TiB	2 x 900
X1	3.840 TiB	2 x 920
X1e	3.840 TiB	2 x 1920

The data on NVMe instance storage is encrypted using an XTS-AES-256 block cipher implemented in a hardware module on the instance. The encryption keys are generated using the hardware module and are unique to each NVMe instance storage device. All

encryption keys are destroyed when the instance is stopped or terminated and cannot be recovered. You cannot disable this encryption and you cannot provide your own encryption key[1].

NVMe on EC2 Instances

Amazon EC2 instances based on the Nitro hypervisor feature local NVMe SSD storage and also expose Amazon Elastic Block Store (Amazon EBS) volumes as NVMe block devices. This is why certain operating system levels are required for Nitro based instances. In other words, only an AMI that has the required NVMe drives installed allows you to launch a Nitro based instance. See *AMI and Operating System* for instructions on verifying that your AMI will run on a Nitro based instance.

If you use EBS volumes on Nitro based instances, configure two kernel settings to ensure optimal performance. Refer to the NVMe EBS Volumes page of the *Amazon Elastic Compute Cloud User Guide for Linux Instances* for more information.

Amazon Elastic File System (Amazon EFS)

You can opt for building your own NFS file server on AWS (discussed in the "Traditional NFS File System" section), or you can launch a shared NFS file system using Amazon Elastic File System (Amazon EFS). Amazon EFS provides simple, scalable NFS based file storage for use with Amazon EC2 instances in the AWS Cloud. A fully managed, petabyte scale file system, 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, increasing and decreasing 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. The data stored on an EFS file system 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 shared file systems that do not span multiple Availability Zones. Because of this, it is important to test your workloads at scale to ensure EFS meets your performance requirements.

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 offerings include following range of storage classes.

- **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.
- **Amazon Glacier** for long-term data archival.

Amazon S3 also offers configurable lifecycle policies for managing your objects so that they are stored cost effectively throughout their lifecycle.

Amazon S3 is accessed via HTTP REST requests typically through the AWS software development kits (SDKs) or the AWS Command Line Interface (AWS CLI). You can use the AWS CLI to copy data to and from Amazon S3 in the same way that you copy data to other remote file systems, using `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. For example, you can quickly and efficiently copy data from Amazon S3 to Amazon EC2 instances and Amazon EBS storage 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 the runtime of a performance critical application. The best architectures for high performance, data intensive computing available on AWS consist of Amazon S3, Amazon EC2, Amazon EBS, and Amazon EFS to balance performance, durability, scalability, and cost for each specific application.

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 option 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 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 throughput.

The largest Amazon EC2 instances support 25 Gbps of network bandwidth and up to 80,000 IOPS and 1,750 MB/s to Amazon EBS. If the data is temporary or scratch data, you can use an instance with NVMe volumes to optimize the backend storage. For example, you can use the i3.16xlarge with 8 NVMe volumes that is capable of up to 16GB/s and 3M IOPS for local access. The 25 Gbps network connection to the i3.16xlarge then becomes the bottleneck, and not the backend storage. This setup results in an NFS that is capable of 2.5 GB/s.

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 way, 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. Then, you can delete the storage server, thereby saving additional cost.

When 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 to provide for fault tolerance.

Although out of the scope of this document, if you want more automated and managed solutions, consider AWS partner storage solutions. Examples of partners that provide solutions for running high performance storage on AWS include SoftNAS, WekaIO, and NetApp.

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 the workflows to reduce or eliminate the need for traditional shared storage systems. Cloud-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, and so on) 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 can provide substantial performance and reliability benefits, in particular for frontend RTL batch regressions.

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 (such as 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 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.

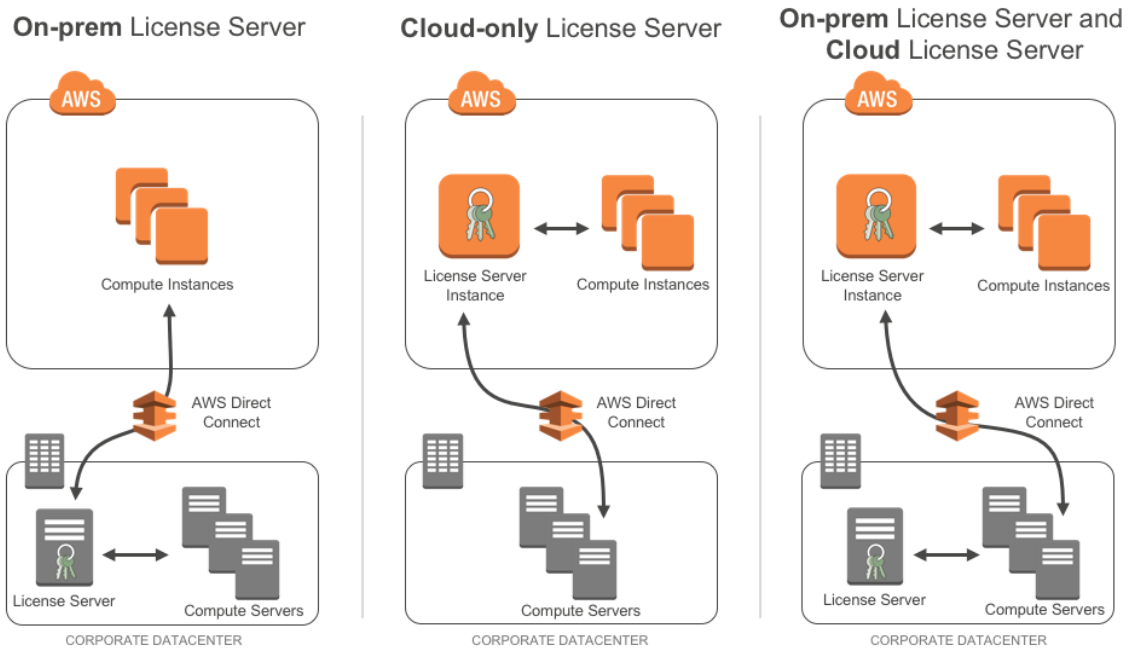


Figure 4: License server deployment scenarios

Licensed applications are sometimes sensitive to network latency and jitter between the execution host and the license server. Although internet-based VPN is often a good choice for connecting to AWS from your corporate datacenter, network latency over the Internet can vary, affecting performance and reliability of some licensed applications. Alternatively, a private, dedicated connection from your on-premises network to the nearest AWS Region using AWS Direct Connect can provide a reliable network connection with consistent latency.

Improving License Server Reliability

License servers are critical components in almost any EDA computing infrastructure. A loss of license services can bring engineering work to a halt across the enterprise. Hosting licenses in the AWS Cloud can provide improved reliability of license services with the use of a floating elastic network interface (ENI). These ENIs have a fixed, immutable MAC address that can be associated with software license keys.

The implementation of this high availability solution begins with the creation of an ENI that is attached to a license server instance. Your license keys are associated with this network interface. If a failure is detected on this instance, you, or your custom automation, can detach the ENI and attach it to a standby license server. Because the

ENI maintains its IP and MAC addresses, network traffic begins flowing to the standby instance as soon as you attach the network interface to the replacement instance.

This unique capability enables license administrators to provide a level of reliability that can be difficult to achieve using on-premises servers in a traditional datacenter. This is another example of the benefits of the elastic and programmable nature of the cloud.

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 software as a service (SaaS), platform as a service (PaaS), customer self-managed, and bring your own license (BYOL) models. In the semiconductor sector, AWS works closely with major vendors of EDA software to help optimize performance, scalability, cost, and application security. 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.

Remote Desktops

While the majority of EDA workloads are executed as batch jobs (see *Orchestration*), EDA users may at times require direct console access to compute servers, or use applications that are graphical in nature. For example, it might be necessary to view waveforms or step through a simulation to identify and resolve RTL regression errors, or it might be necessary to view a 2D or 3D graphical representation of results generated during signal integrity analysis. Some applications, such as printed circuit layout software, are inherently interactive and require a high quality, low latency user experience.

There are multiple ways to deploy remote desktops for such applications on AWS. You have the option of using open-source software such as Virtual Network Computing (VNC), or commercial remote desktop solutions available from AWS partners. You can

also make use of AWS solutions including NICE desktop cloud visualization (NICE DCV) and Amazon WorkSpaces.

NICE DCV

NICE Desktop Cloud Visualization is a remote visualization technology that enables users to securely connect to graphic-intensive 3D applications hosted on an Amazon EC2 instance. With NICE DCV, you can provide high-performance graphics processing to remote users by creating secure client sessions. This enables your interactive EDA users to use resource-intensive applications with relatively low-end client computers by using one or more EC2 instances as remote desktop servers, including GPU acceleration of graphics rendered in the cloud.

In a typical NICE DCV scenario for EDA, a graphic-intensive application, such as a 3D visualization of an electromagnetic field simulation, or a complex, interactive schematic capture session, is hosted on a high-performance EC2 instance that provides a high-end GPU, fast I/O capabilities, and large amounts of memory.

The NICE DCV server software is installed and configured on a server (an EC2 instance) and it is used to create a secure session. You use a NICE DCV client to remotely connect to the session and use the application hosted on the server. The server uses its hardware to perform the high-performance processing required by the hosted application. The NICE DCV server software compresses the visual output of the hosted application and streams it back to you as an encrypted pixel stream. Your NICE DCV client receives the compressed pixel stream, decrypts it, and then outputs it to your local display.

NICE DCV was specifically designed for high performance technical applications, and is an excellent choice for EDA, in particular if you are using Red Hat Enterprise Linux or CentOS operating systems on your remote desktop environment. NICE DCV also supports modern Linux desktop environments including modern Linux desktops such as Gnome 3 on RHEL 7.

NICE DCV uses the latest NVIDIA Grid SDK technologies, such as NVIDIA H.264 hardware encoding, to improve performance and reduce system load. NICE DCV also supports lossless quality video compression when the network and processor conditions allow, and it automatically adapts the video compression levels based on the network's available bandwidth and latency.

Amazon Workspaces

Amazon WorkSpaces is a managed, secure cloud desktop service. You can use Amazon WorkSpaces to provision either Windows or Linux desktops in just a few minutes and quickly scale to provide thousands of desktops to workers across the globe. You can pay either monthly or hourly, just for the WorkSpaces you launch, which helps you save money when compared to traditional desktops and on-premises virtual desktop infrastructure (VDI) solutions. Amazon WorkSpaces helps you eliminate the complexity in managing hardware inventory, OS versions and patches, and VDI, which helps simplify your desktop delivery strategy. With Amazon WorkSpaces, your users get a fast, responsive desktop of their choice that they can access anywhere, anytime, from any supported device.

Amazon WorkSpaces offers a range of CPU, memory, and solid-state storage bundle configurations that can be dynamically modified so you have the right resources for your applications. You don't have to waste time trying to predict how many desktops you need or what configuration those desktops should be, helping you reduce costs and eliminate the need to over-buy hardware.

Amazon WorkSpaces is an excellent choice for organizations wanting to centrally manage remote desktop users and applications, and for users that can make use of Windows or Amazon Linux 2 for the remote desktop environment.

User Authentication

User authentication is covered in more detail in the *Security and Governance in the AWS Cloud* section, but AWS offers several options for connecting with an on-premises authentication server, migrating users to AWS, or architecting an entirely new authentication solution.

Orchestration

Orchestration refers to the dynamic management of compute and storage resources in an EDA cluster, as well as the management (scheduling and monitoring) of individual jobs being processed in a complex workflow, for example during RTL regression testing or IP characterization. For these and many other typical EDA workflows, the efficient use of compute and storage resources—as well as the efficient use of EDA software licenses—depends on having a well-orchestrated, well-architected batch computing environment.

EDA workload management gains new levels of flexibility in the cloud, making resource and job orchestration an important consideration for your workload. AWS provides a range of solutions for workload orchestration: fully-managed services enable you to focus more on job requests and output over provisioning, configuring and optimizing the cluster and job scheduler, while self-managed solutions enable you to configure and maintain cloud-native clusters yourself, leveraging traditional job schedulers to use on AWS or in hybrid scenarios.

Describing all possible methods of orchestration for EDA is beyond the scope of this document; however, it is important to know that the same orchestration methods and job scheduling software used in typical, legacy EDA environments can also be used on AWS. For example, commercial and open-source job scheduling software can be migrated to AWS, and be enhanced by the addition of Auto Scaling (for dynamic resizing of EDA clusters in response to demand or other triggers), CloudWatch (for monitoring the compute environment, for example CPU utilization and server health), and other AWS services to increase performance and security, while reducing costs.

CfnCluster

CfnCluster (cloud formation cluster) is a framework that deploys and maintains high performance computing clusters on Amazon Web Services (AWS). Developed by AWS, CfnCluster facilitates both quick start proof of concepts (POCs) and production deployments. CfnCluster supports many different types of clustered applications, including EDA, and can easily be extended to support different frameworks.

CfnCluster integrates easily with existing job scheduling software, and can automatically launch servers in response to queue depths and other triggers. CfnCluster is also able to launch shared file systems, cluster head nodes, license servers, and others resources. CfnCluster is open-source and easily extensible for your unique workflow requirements.

AWS Batch

AWS Batch is a fully-managed service that enables you to easily run large-scale compute workloads on the cloud, including EDA jobs, without having to worry about resource provisioning or managing schedulers. Interact with AWS Batch via the web console, AWS CLI, or SDKs. AWS Batch is an excellent alternative for managing massively parallel workloads.

EnginFrame

EnginFrame is an HPC portal that can be deployed on the cloud, or on-premise. EnginFrame is integrated with a wide range of open source and commercial batch scheduling systems, and is a one-stop-shop for job submission, control and data management.

All of the preceding options (CfnCluster, AWS Batch, and EnginFrame), as well as partner-provided solutions, are being successfully deployed by EDA users on AWS. Discuss your specific orchestration needs with an AWS technical specialist.

Optimizing EDA Tools 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 run times 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.

To this point, we have covered the solution components for your architecture on AWS. Now, in an effort to be more prescriptive, we present specific recommendations and configuration parameters that should help you achieve expected performance for your EDA tools.

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.

Amazon EC2 Instance Types

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

Table 4: EDA tools and corresponding instance type

Instance Name	*Max Core Count	CPU Clock Frequency	Max Total RAM in GiB & (GiB/core)	Local NVMe	Typical EDA Application
Z1d	24	4.0 GHz	384 (16)	Y	Formal verification RTL Simulation Batch RTL Simulation Interactive RTL Gate Level Simulation
R5 / R5d	48	Up to 3.1 GHz	768 (16)	Y (R5d)	RTL Simulation Multi-Threaded
R4	32	2.3 GHz	488 (15.25)		RTL Simulation Multi-Threaded Place & Route
M5 / M5d	48	Up to 3.1 GHz	384 (16)	Y (M5d)	Remote Desktop Sessions
C5 / C5d	36	Up to 3.5 GHz	144 (4)	Y (C5d)	RTL Simulation Interactive RTL Gate Level Simulation
X1	64	2.3 GHz	1,952 (30.5)	Y	Place & Route Static Timing Analysis
X1e	64	2.3 GHz	3,904 (61)	Y	Place & Route Static Timing Analysis
C4	18	2.9 GHz (boost to 3.5)	60 (3.33)		Formal verification RTL Simulation Interactive

*NOTE: AWS uses vCPU (which is an Intel Hyper-Thread) to denote processors, for this table we are using cores.

Operating System Optimization

After you have 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

If you are running a Nitro based instance, you need to use certain operating system levels. If you run a Xen based instance instead, you should still use one of these OS levels for EDA workloads (specifically required for ENA and NVMe drivers):

- Amazon Linux or Amazon Linux 2
- CentOS 7.4 or 7.5
- Red Hat Enterprise Linux 7.4 or 7.5

Disable Hyper-Threading

On current generation Amazon EC2 instance families (other than the T2 instance family), AWS instances have Intel Hyper-Threading Technology (HT Technology) enabled by default. You can disable HT Technology if you determine that it has a negative impact on your application's performance.

You can run this command to get detailed information about each core (physical core and Hyper-Thread):

```
$ cat /proc/cpuinfo
```

To view cores and the corresponding online Hyper-Threads, use the `lscpu –extended` command. For example, consider the Z1d.2xlarge, which has 4 cores with 8 total Hyper-Threads. If you run the `lscpu –extended` command before and after disabling Hyper-Threading, you can see which threads are online and offline:

```
$ lscpu --extended
CPU NODE SOCKET CORE L1d:L1i:L2:L3 ONLINE
0   0    0      0    0:0:0:0       yes
1   0    0      1    1:1:1:0       yes
2   0    0      2    2:2:2:0       yes
3   0    0      3    3:3:3:0       yes
4   0    0      0    0:0:0:0       yes
5   0    0      1    1:1:1:0       yes
6   0    0      2    2:2:2:0       yes
7   0    0      3    3:3:3:0       yes

$ ./disable_ht.sh

$ lscpu --extended
CPU NODE SOCKET CORE L1d:L1i:L2:L3 ONLINE
0   0    0      0    0:0:0:0       yes
1   0    0      1    1:1:1:0       yes
2   0    0      2    2:2:2:0       yes
3   0    0      3    3:3:3:0       yes
4   -    -      -    :::           no
5   -    -      -    :::           no
6   -    -      -    :::           no
7   -    -      -    :::           no
```

Another way to view the vCPUs pairs (that is, Hyper-Threads) of each core is to view the `thread_siblings_list` for each core. This list shows two numbers that indicate Hyper-Threads for each core. To view all thread siblings, you can use the following command, or substitute "*" with a CPU number:

```
$ cat/sys/devices/system/cpu/cpu*/topology/thread_siblings_list | sort -un
0,4
1,5
2,6
3,7
```

Disable HT Using the AWS feature - CPU Options

To disable Hyper-Threading using CPU Options, use the AWS CLI with `run-instances` and the `--cpu-options` flag. The following is an example with the Z1d.12xlarge:

```
$ aws ec2 run-instances --image-id ami- asdfasdfasdfasdf \
   --instance-type z1d.12xlarge --cpu-options \
   "CoreCount=24,ThreadsPerCore=1" --key-name My_Key_Name
```

To verify the CpuOptions were set, use `describe-instances`:

```
$ aws ec2 describe-instances --instance-ids i-1234qwer1234qwer
...
"CpuOptions": {
     "CoreCount": 24,
     "ThreadsPerCore": 1
},
...
```

Disable HT on a Running System

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. See the following example.

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

Disable HT Using the Boot File

You can also 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.

For example, the `maxcpus` value for a Z1d.12xlarge instance is 24 to disable Hyper-Threading:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8
 net.ifnames=0 biosdevname=0 nvme_core.io_timeout=4294967295 maxcpus=24
```

Refer to Appendix C – Updating the Linux Kernel Command Line for instructions on updating the kernel command line.

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

Change Clocksource to TSC

On previous generation instances that are using the Xen hypervisor, consider updating the clocksource to TSC, as the default is the Xen pvclock (which is in the hypervisor). To avoid communication with the hypervisor and use the CPU clock instead, use `tsc` as the clocksource.

The `tsc` clocksource is not supported on Nitro instances. The default `kvm-clock` clocksource on these instance types provides similar performance benefits as `tsc` on previous-generation Xen based instances.

To change the clocksource on a Xen based instance , run the following command:

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

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

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

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 below:

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

Limiting Deeper C-states (Sleep State)

C-states control the sleep levels that a core may enter when it is inactive. You may want to control C-states to tune your system for latency versus performance. Putting cores to sleep takes time, and although a sleeping core allows more headroom for another core to boost to a higher frequency, it takes time for that sleeping core to wake back up and perform work.

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8 net.ifnames=0
  biosdevname=0 nvme_core.io_timeout=4294967295 intel_idle.max_cstate=1"
```

Refer to Appendix C – Updating the Linux Kernel Command Line for instructions on updating the kernel command line.

For more information about Amazon EC2 instance processor states, refer to the Processor State Control for Your EC2 Instance page in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.

Enable Turbo Mode (Processor State) on Xen Based Instances

For our current Nitro based instance types, you cannot change turbo mode, as this is already set to the optimized value for each instance.

If you are running on a Xen based instance that is using an entire socket or multiple sockets (for example, r4.16xlarge, r4.8xlarge, c4.8xlarge) you can take advantage of the turbo frequency boost, especially if you have disabled HT Technology.

Amazon Linux and Amazon Linux 2 have 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, refer to the Processor State Control for Your EC2 Instance page in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.

Change to Optimal Spinlock Setting on Xen Based Instances

For the instances that are using the Xen hypervisor (not Nitro), you should update the spinlock setting. Amazon Linux, Amazon Linux 2, 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. Some EDA tools are not optimized for multi-core and consequently rely heavily on spinlocks. Accordingly, we recommend that EDA customers disable paravirtualized spinlock on EC2 instances.

To disable the paravirtualized mode of spinlock on a Xen based instnace, add `xen_nopvspin=1` to the kernel command line in `/boot/grub/grub.conf` and restart. The 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
```

Refer to Appendix C – Updating the Linux Kernel Command Line for instructions on updating the kernel command line.

Networking

AWS Enhanced Networking

Make sure to use enhanced networking for all instances, which is a requirement for launching our current Nitro based instances. For more information about enhanced networking, including build and install instructions, refer to the Enhanced Networking on Linux page in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.

Cluster Placement Groups

A cluster placement group is a logical grouping of instances within a single Availability Zone. Cluster placement groups provide non-blocking, non-oversubscribed, fully bisectional connectivity. In other words, all instances within the placement group can communicate with all other nodes within the placement group at the full line rate of 10 Gpbs flows and 25 Gpbs aggregate without any slowing due to over-subscription. For more information about placement groups refer to the Placement Groups page in the *Amazon Elastic Compute Cloud User Guide for Linux Instances*.

Verify Network Bandwidth

One method to ensure you are configuring ENA correctly is to benchmark the instance to instance network performance with iperf3. Refer to Network Throughput Benchmark Linux EC2 for more information.

Storage

Amazon EBS Optimization

Make sure to choose your instance and EBS volumes to suit the storage requirements for your workloads. Each EC2 instance type has an associated EBS limit, and each EBS volume type has limits as well. For example, the m4.16xlarge instance type has a io1 volume type with a maximum throughput of 500MB/s.

NFS Configuration and Optimization

Prior to setting up an NFS server on AWS, you need to enable Amazon EC2 enhanced networking. We recommend using Amazon Linux 2 for your NFS server AMI.

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 in following example:

```
$ 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, choose the correct instance size and number of EBS volumes. Within a single family, 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. Refer to Appendix A – Optimizing Storage for more information and diagrams for building an NFS server on AWS.

Kernel Virtual Memory

Typical operating system distributions are not tuned for large machines like those offered by AWS for EA 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
```

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 achieve with traditional on-premises environments. This section outlines some of the ways you can protect data in the AWS Cloud.

Isolated Environments for Data Protection and Sovereignty

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 do not have any direct access to the internet. Users who need to access the environment must first connect to the Bastian Node (also 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.

Often, secure FTP is required in isolated environments. Organizations commonly use secure FTP to download tools from vendors, copy completed designs to fabrication facilities, or to update IP from suppliers. 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. 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 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 do for any remote data center configuration.

If your organization wants to run 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.[2] 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 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 to enable inbound and outbound filtering at the instance level and subnet level, respectively. 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.

Network access control lists (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, refer to the Security page in the Amazon Virtual Private Cloud User Guide.

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 transit 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. In addition, it 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 in transit 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.

EC2 Instance Store Encryption

The data on NVMe instance storage is encrypted using an XTS-AES-256 block cipher implemented in a hardware module on the instance. The encryption keys are generated using the hardware module and are unique to each NVMe instance storage device. All encryption keys are destroyed when the instance is stopped or terminated and cannot be

recovered. You cannot disable this encryption and you cannot provide your own encryption key.[1]

Amazon S3 Encryption

When you use encryption with Amazon S3, Amazon S3 encrypts your data at the object level. 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 can be used to enforce encryption of all incoming S3 objects.

Because access to Amazon S3 is provided over HTTP endpoints, you can also leverage bucket policies to ensure that all data transfer in and out occurs over a TLS connection to guarantee 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, refer to the AWS IAM User Guide.

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, refer to the Amazon CloudWatch User Guide.

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, refer to the Amazon CloudWatch Log User Guide.

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, refer to the AWS CloudTrail User Guide.

Amazon Macie – Amazon Macie is a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Amazon Macie recognizes sensitive data such as personally identifiable information (PII) or intellectual property, and provides you with dashboards and alerts that give visibility into how this data is being accessed or moved. The fully managed service continuously monitors data access activity for anomalies, and generates detailed alerts when it detects risk of unauthorized access or inadvertent data leaks.

AWS GuardDuty – Amazon GuardDuty is a threat detection service that continuously monitors for malicious or unauthorized behavior to help you protect your AWS accounts and workloads. It monitors for activity such as unusual API calls or potentially unauthorized deployments that indicate a possible account compromise. GuardDuty also detects potentially compromised instances or reconnaissance by attackers.

AWS Shield – AWS Shield is a managed Distributed Denial of Service (DDoS) protection service that safeguards applications running on AWS. AWS Shield provides always-on detection and automatic inline mitigations that minimize application downtime and latency, so there is no need to engage AWS Support to benefit from DDoS protection.

AWS Config – Use to assess, audit, and evaluate the configurations of your AWS resources. AWS 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, refer to the AWS Config Developer Guide.

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, refer to the AWS Organizations User Guide.

AWS Service Catalog – AWS Service Catalog allows organizations to create and manage catalogs of IT services that are approved for use on AWS. These IT services can include everything from virtual machine images, servers, software, and databases to complete multi-tier application architectures. AWS Service Catalog allows you to centrally manage commonly deployed IT services, and helps you achieve consistent governance and meet your compliance requirements, while enabling users to quickly deploy only the approved IT services they need.

Contributors

The following individuals contributed to this document:

- Mark Duffield, Worldwide Tech Leader, Semiconductors, Amazon Web Services
- David Pellerin, Principal Business Development for Infotech/Semiconductor, Amazon Web Services
- Matt Morris, Senior HPC Solutions Architect, Amazon Web Services
- Nafea Bshara, VP/Distinguished Engineer, Amazon Web Services

Document Revisions

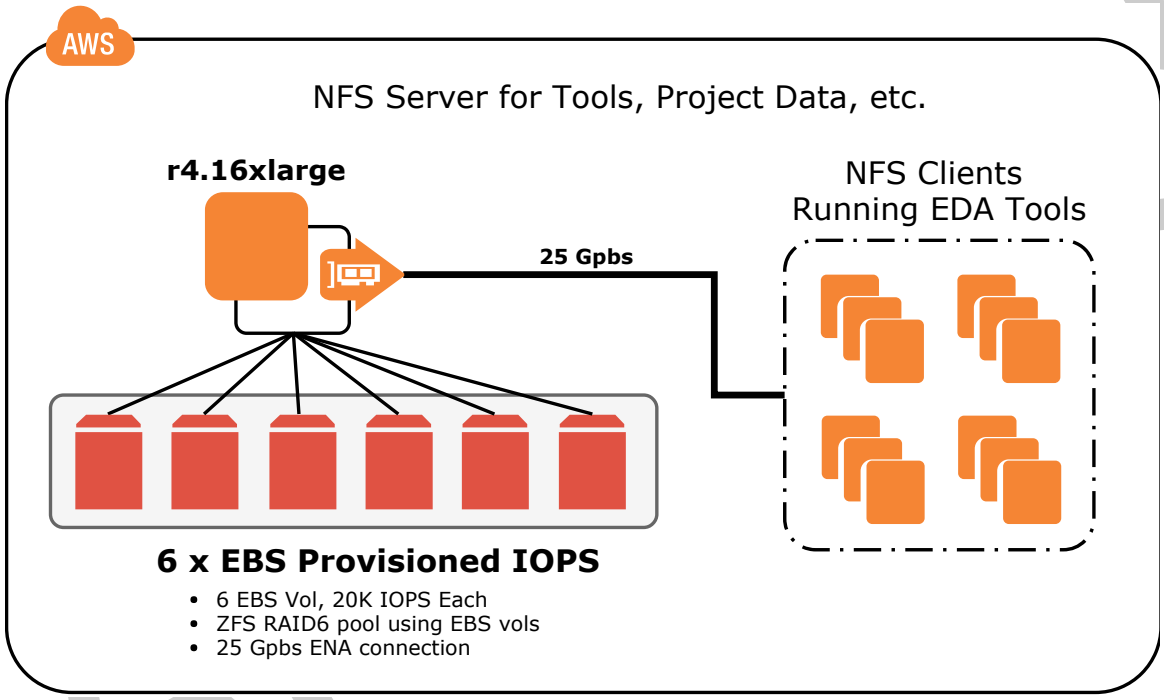
Date	Description
September 2018	2018 update
October 2017	First publication

Appendix A – Optimizing Storage

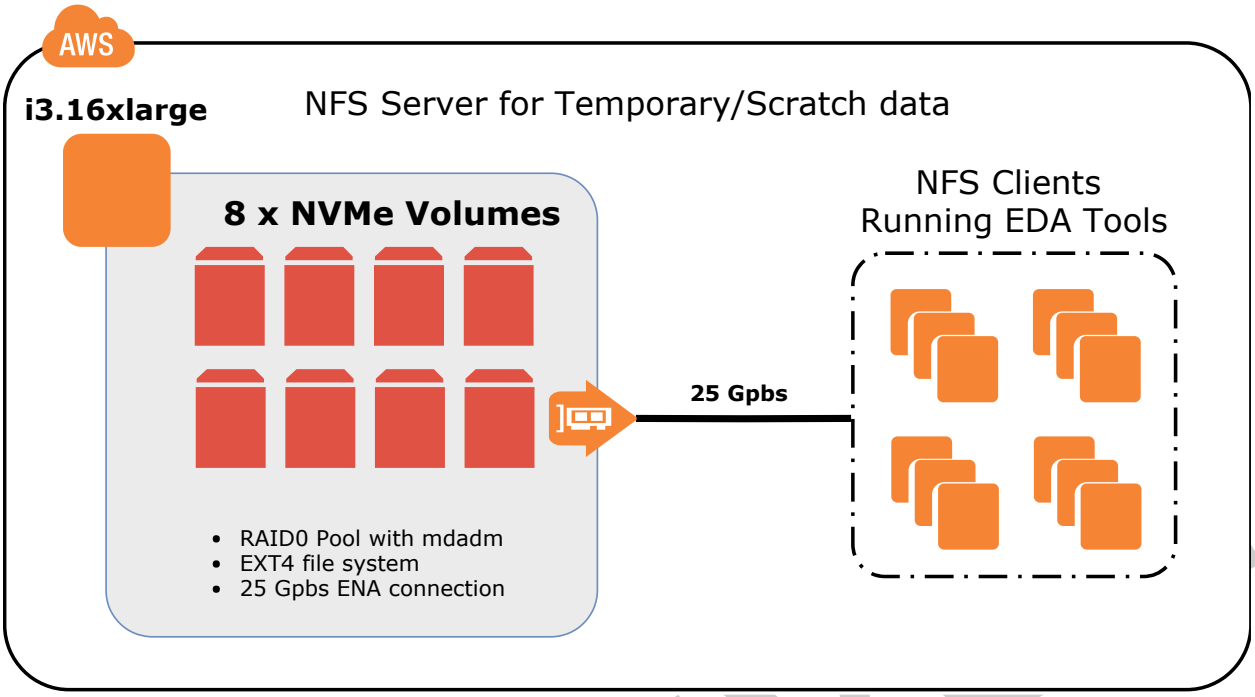
There are many storage options on AWS, and some have already been covered at a high level. As semiconductor workloads rely on shared storage, building an NFS server may be the first step to running EDA tools. This section includes two possible NFS architectures that can achieve suitable performance for most workloads.

NFS Storage

NFS server capable of 1.75 GB/s with 75,000 IOPS



NFS server capable of 2.5 GB/s and > 100,000 IOPS



Appendix B – Reference Architecture

The following diagram represents a common architecture for an elastic EDA computing environment in AWS. This design provides the following key infrastructure components:

- Amazon EC2 AutoScaling Group for elasticity
- AWS Direct Connect for dedicated connectivity to AWS
- Amazon Linux WorkSpaces for remote desktops
- Amazon EC2 based compute, license, and scheduler instances
- Amazon EC2 based NFS servers and Amazon EFS for sharing file systems between compute instances

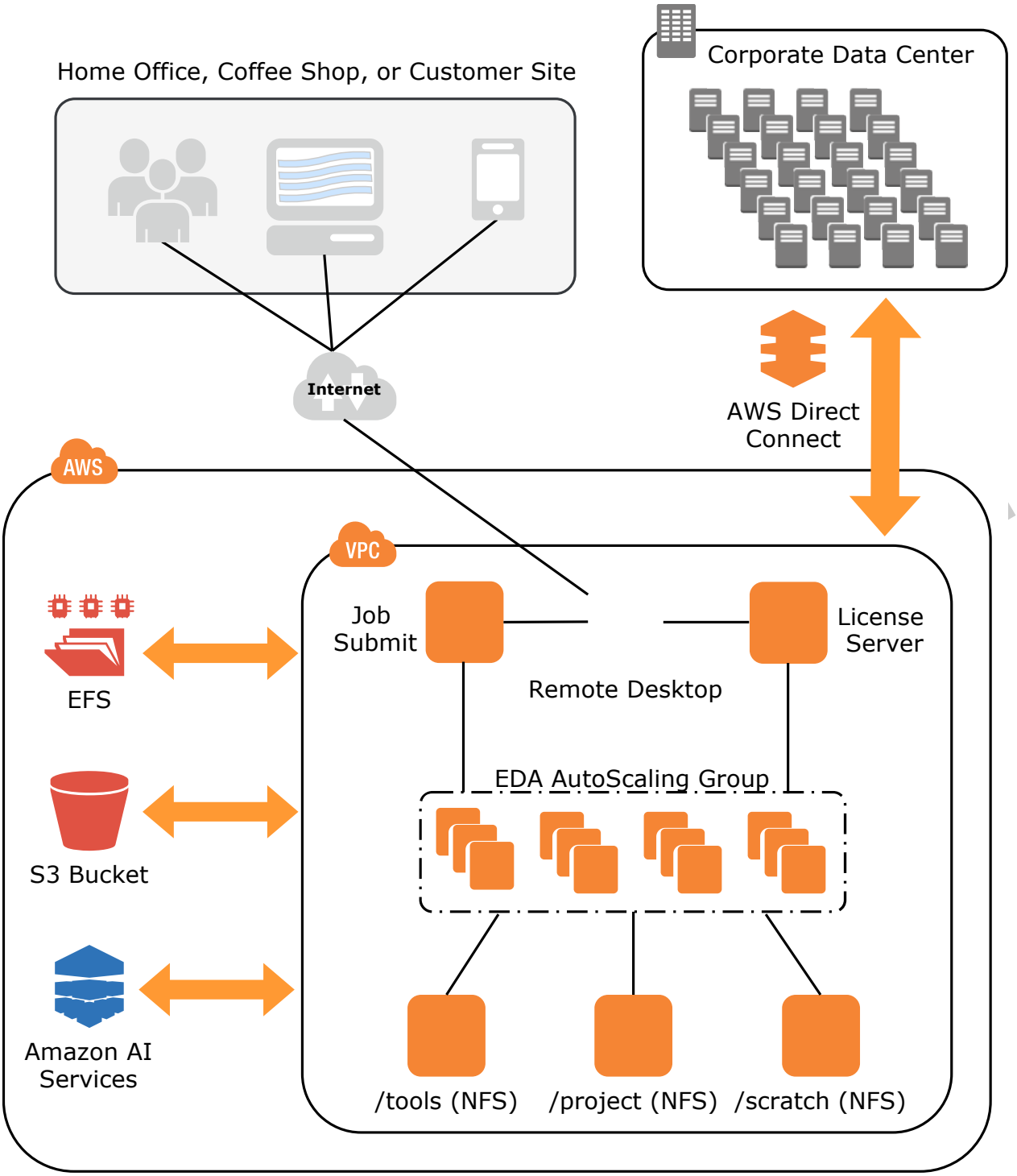


Figure 5: EDA architecture on AWS

Appendix C – Updating the Linux Kernel Command Line

Update a system with /etc/default/grub file

1. Open the /etc/default/grub file with your editor of choice.

   ```
   $ sudo vim /etc/default/grub
   ```

2. Edit the GRUB_CMDLINE_LINUX_DEFAULT line, and make necessary changes. For example:

```
GRUB_CMDLINE_LINUX_DEFAULT="console=tty0 console=ttyS0,115200n8
 net.ifnames=0 biosdevname=0 nvme_core.io_timeout=4294967295
 intel_idle.max_cstate=1"
```

3. Save the file and exit your editor.
4. Run the following command to rebuild the boot configuration.

   ```
   $ grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

5. Reboot your instance to enable the new kernel option.

   ```
   $ sudo reboot
   ```

Update a system with /boot/grub/grub.conf file

1. Open the /boot/grub/grub.conf file with your editor of choice.

   ```
   $ sudo vim /boot/grub/grub.conf
   ```

2. Edit the kernel line, for example (some info removed for clarity)

```
# 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-ver.amzn1.x86_64 <other_info> intel_idle.max_cstate=1
initrd /boot/initramfs-3.14.26-24.46.amzn1.x86_64.img
```

3. Save the file and exit your editor.
4. Reboot your instance to enable the new kernel option.

   ```
   $ sudo reboot
   ```

Verify Kernel Line

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

```
$ 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
```

Notes

[1] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ssd-instance-store.html

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