AWS for Industries

Crypto market-making latency and Amazon EC2 shared placement groups

by Atiek Arian, Boris Litvin, and Severin Gassauer-Fleissner | on 24 JAN 2023 | in Financial Services, Industries |
Permalink | Share

 𝖿   𝕏   in   ✉   https://aws.amazon.com/blogs/industries/crypto-market-ma

Crypto currencies started as an experiment in 2008, initially supported by global developer communities and retail traders. They have since expanded into a multi-billion dollar industry, traded on numerous Crypto exchanges with growing amounts of institutional interest. Many of these exchanges have been built on AWS, resulting in the opportunity for High Frequency Traders (HFTs) to also build and optimize their own platforms on the AWS Cloud. Recently, we launched Amazon Elastic Compute Cloud (Amazon EC2) shared cluster placement groups (CPGs), which enables Crypto exchanges to extend their CPGs to HFT customers in different AWS accounts. This improves latency between exchange matching and HFT trading engines, delivering tick-to-trade latency advantages that HFTs have become accustomed to in traditional colocation facilities.

The typical strategies employed by HFTs are market-making and arbitrage. A market-making strategy aims to capture spread between the buy and sell price of a security, called the bid-ask spread. Exchanges depend on market-makers to supply bids and offers, collectively referred to as liquidity, which attracts customers who want to invest in the associated securities. Arbitrage strategies exploit the price discrepancies of the same security in different markets – there are multiple forms of arbitrage, but this is the simplest. These strategies play an important role in the industry – market-making makes sure that securities are tradable by providing liquidity, while arbitrage maintains fair prices. The effective execution of these strategies is heavily dependent on latency, which impacts the profitability of market-makers. Consequently, exchanges with better latency profiles attract more and better-quality liquidity, in turn attracting customers and higher trading volumes.

Concepts around optimizing network latency for market-making aren't new. They have been well understood and applied in equities trading for decades. A large portion of trades in the traditional equity markets are executed and cancelled within the first 1,000 microseconds of the order being placed, with the bulk of activity occurring in the first 100 microseconds. Market-makers must be the first to respond to mitigate market risks. These principles apply equally to Crypto markets and improved latency is one source of increased profitability. Although the principles remain the same, the trading venues are different and require an understanding of how to optimize latency within the AWS Cloud as opposed to a traditional physical colocation facility.

Optimizing network latency is a large topic with multiple dimensions. However, one key aspect to improving network performance between two points on a network is to decrease the physical distance between them. In the context of this discussion, these points are the Crypto exchange matching engine and a Crypto market-making trading engine. In this post, we discuss how to use Amazon EC2 shared CPGs to reduce this distance, and allow exchanges and market-makers to achieve greater network proximity on the AWS Cloud. We also summarize the performance improvements provided by CPGs and some considerations around Amazon Virtual Private Cloud (Amazon VPC) logical connectivity options for these use-cases.

Amazon EC2 cluster placement groups

Amazon EC2 placement groups are a feature provided by Amazon EC2 that allows customers to control the placement of instances across the underlying hardware within an Availability Zone (AZ). Although three different types of placement group are available (cluster, partition, and spread), cluster placement groups are the most relevant in the context of optimizing network latency. CPGs are a way of improving network proximity for Amazon EC2 instances. This reduces the latency for traffic traversing the network between members contained within the group. The network infrastructure within an AZ is segregated into cells, with racks providing instance capacity connected via a series of layers to specific network cells in the AZ. The following diagram illustrates this at a high level for an example AWS region and a single AZ (most AWS regions have three or more AZs). Further details are shown in this AWS re:Invent 2019 talk:
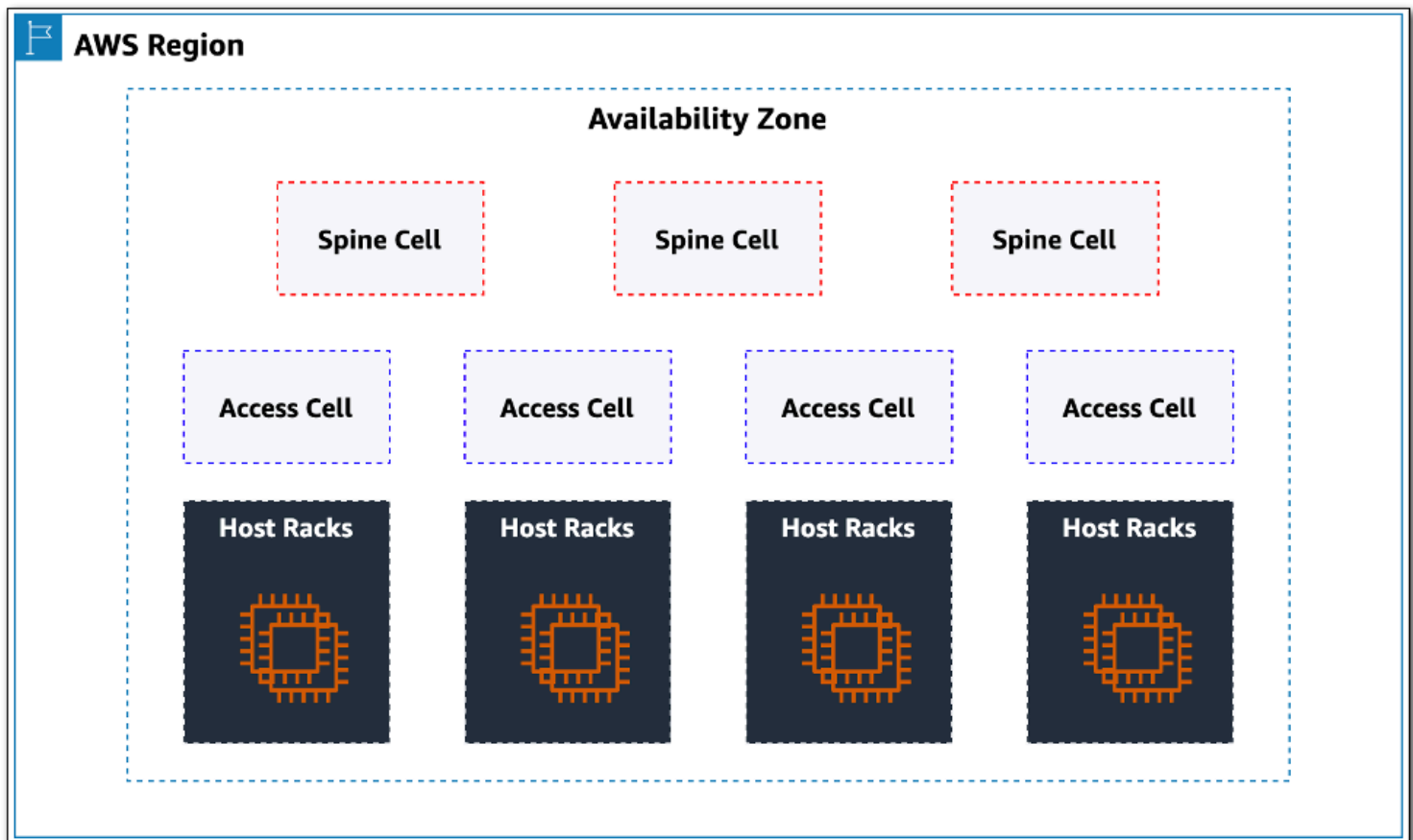
When customers provision Amazon EC2 instances without a placement group, these instances can be provisioned from capacity distributed across multiple spine cells in the AZ. CPGs provide a placement strategy that allows Amazon EC2 instances to share connectivity to the same spine cell, thereby making sure of lower latency on the underlying AWS network. Conceptually, this can be viewed as a placement group network, within which the instances can participate, as illustrated here:
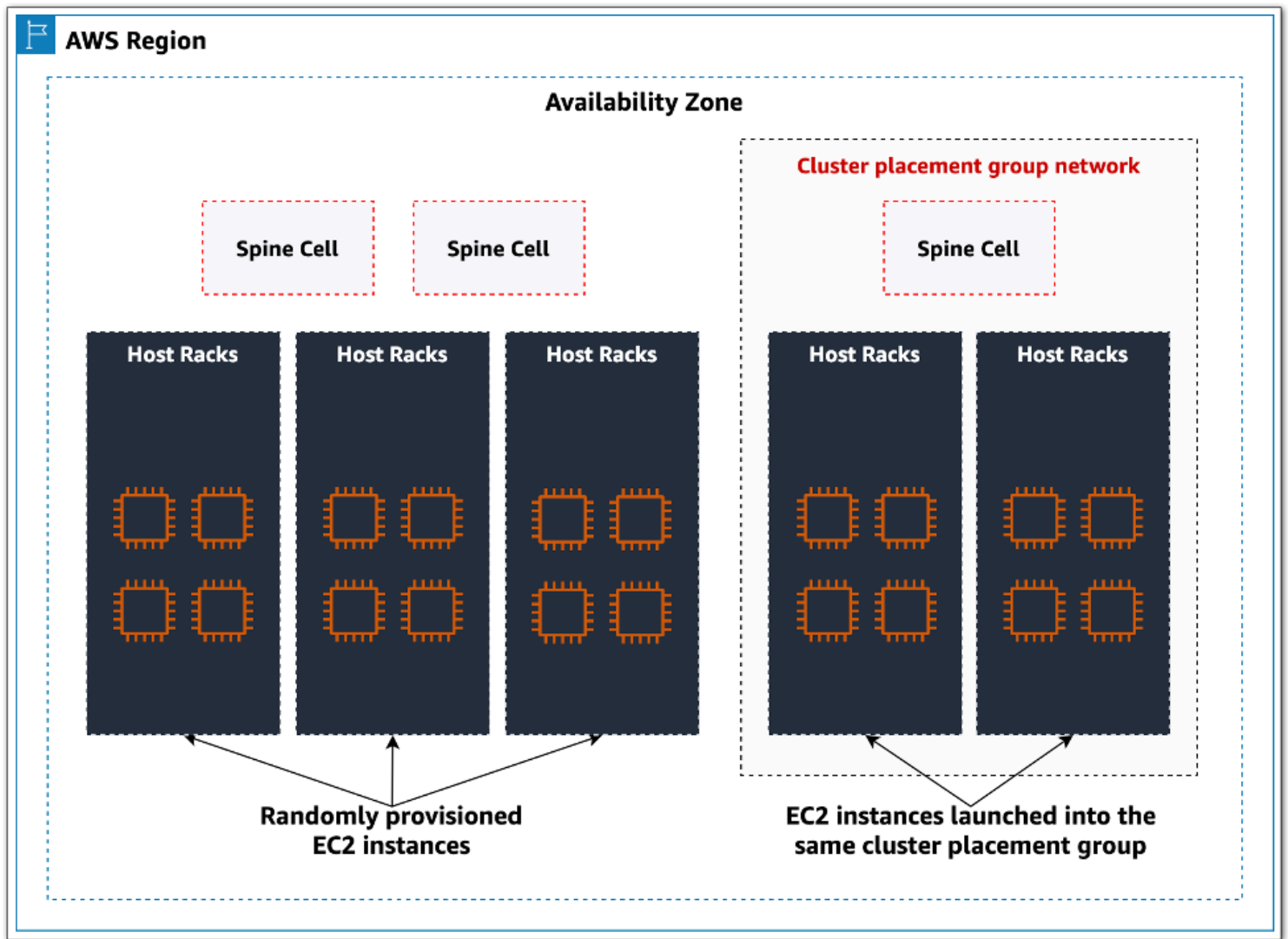
Since instances within a CPG are located in the same high-bisection bandwidth network segment, they enjoy higher per-flow throughput limits of up to 10Gb/s bandwidth per flow, as well as a reduction in latency and increased packet processing rates. We discuss how this can benefit Crypto market-making later in this post.

Up until October 2022, placement groups couldn't be shared between AWS accounts. This presented an adoption challenge for market-making use-cases since Crypto exchanges, and associated market-makers, are separate entities operating their application stacks within distinct AWS accounts. With the release of Amazon EC2 shared placement groups, exchanges and market-makers are now free to work together to achieve closer network proximity between themselves within an AZ. Therefore, crypto exchanges can provide faster access to their matching engines, thereby improving liquidity, and market-makers can benefit from better trading performance which results in an enhanced experience for their investor clients.

Working with Amazon EC2 shared cluster placement groups

Customers can now use AWS Resource Access Manager (AWS RAM) to share Amazon EC2 cluster placement groups with other AWS accounts, inside or outside of their AWS Organization, using the AWS RAM console, AWS Command Line Interface (AWS CLI), or the AWS Software Development Kits (AWS SDK).

Shared cluster placement group walkthrough

In this section, we walk through using shared CPGs in an example scenario involving two separate accounts: one belonging to a Crypto exchange (owner account) and another to a Crypto market-maker (receiver account). The premise of this scenario is an exchange that has deployed Amazon EC2 instances within a CPG and would like to provide a market-maker lower latency access to one or more matching engines running on these instances. The following actions can be performed using the AWS Console, AWS CLI, or AWS SDK. However, we'll mostly work with the console during this walkthrough for illustrative purposes.

A note on AZ names and IDs

Since CPGs are designed to make sure of the physical network proximity between EC2 instances, they don't span AZs. This characteristic requires that separate AWS accounts use the same AZ when working within shared CPGs. AWS maps the same physical AZ *randomly* to the AZ name for all of the AWS accounts within an Organization. The owner and receiver accounts should identify the mappings that exist in their accounts between the AZ names and AZ IDs. The latter refer to the same physical AZ across all of the AWS accounts within a Region. Both accounts should use the AZ name that maps to the same AZ ID. These mappings can be easily identified, per AWS region, as follows:

1. Open the AWS RAM console and in the navigation pane select **Resource Access Manager**.

2. View the mappings under **Your AZ ID**.

Owner Account		Receiver Account	
AZ Name	AZ ID	AZ Name	AZ ID
eu-west-1a	euw1-az3	eu-west-1a	euw1-az1
eu-west-1b	euw1-az1	eu-west-1b	euw1-az2
eu-west-1c	euw1-az2	eu-west-1c	euw1-az3

In the example above, the AWS RAM console output is shown from two separate AWS accounts. In this scenario, if the owner account chooses to launch Amazon EC2 instances into a shared CPG using the AZ name eu-west-1a, this corresponds to AZ ID euw-az3. When launching instances into the shared CPG, the receiver account should choose the AZ name eu-west-1c, which corresponds to the same AZ ID, euw-az3, in their account. This makes sure that they target the same physical AZ as the owner.

Create the cluster placement group and launch EC2 instances

The owner account creates a CPG and launches EC2 instances into it:

1. Open the Amazon EC2 console and navigate to **Placement Groups**.

2. Select **Create placement group**.

3. Name the placement group, select **Cluster** from the **Placement strategy** dropdown menu and add any relevant tags if required.

4. Select **Create group**.

5. EC2 instances can now be launched into this CPG inside of the owner account. This can be specified in the
   Amazon EC2 console on the **Launch instance** page under **Advanced details** using the **Placement group**
   dropdown menu. A placement group ID is also listed for identification purposes. This is unique across all AWS
   accounts and can be viewed by the receiver account once the placement group has been shared. The group will
   be listed as "**Shared: No**", even after we have shared it in the next section, since this account owns the cluster
   placement group:

6. Once one or more EC2 instances have been launched, their placement group information can be viewed in the
   Amazon EC2 console on the Instances page under Host and placement group within the Details tab for that
   instance. In addition to the placement group name, the ID is also displayed and you should make note of it since

this should be used by the receiver account for identification purposes when it launches instances into the placement group:
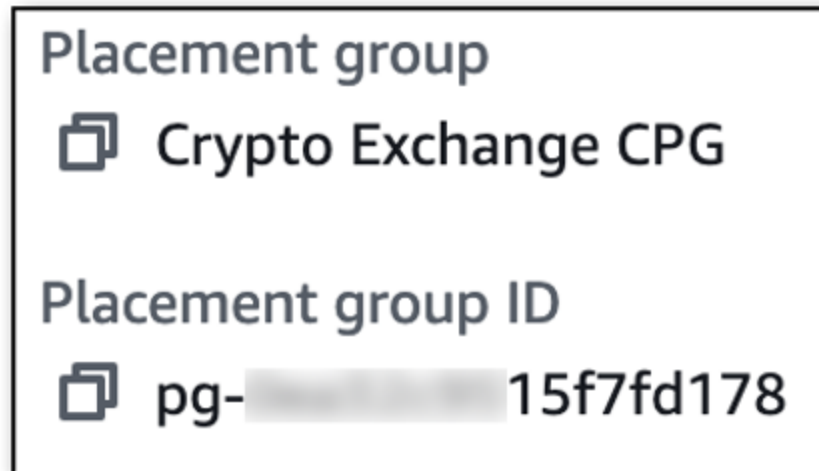
Share the cluster placement group

The owner account can now share the CPG with the receiver account:

1. Open the AWS RAM console and select **Create a resource share**.

2. Name the cluster placement group resource share and select **Placement Groups** from the **Select resource type** dropdown menu.

3. From the list of placement groups select the previously created CPG.

4. Add any relevant tags if required and select **Next**.

5. Review the associated actions granted to principals for the resource share. An *ec2:PlacementGroup* resource type has a single set of available permissions which allow the following actions: *ec2:DescribePlacementGroups*, *ec2:RunInstances*, *ec2:ModifyInstancePlacement* and *ec2:CreateCapacityReservation*. Importantly, among these actions is *ec2:RunInstances*, which will allow principals in the receiver account that you specify to launch EC2 instances into this CPG. Select **Next** to proceed.

6. AWS RAM lets you share placement groups with accounts inside and outside of your Organization. In the context of this use-case, since a Crypto exchange and Crypto market-maker are separate entities not likely to be using AWS accounts within the same Organization, we'll choose to share the resource with a specific account ID. You can also share the placement group with an AWS Identity and Access Management (IAM) role or user within an AWS account. Select the **Allow sharing with anyone** radio button and under the **Principals** section choose **AWS account** from the dropdown menu and enter the receiver AWS account ID. Select **Add** and then **Next**.

7. After reviewing the resource share information, select **Create resource share**.

8. View the details of the resource share. The share will show as **Active** under the **Status** section in the **Summary** box. However, under the **Shared principals** section the **Status** will show as **Associating**. This indicates that the share has been successfully created and is waiting to be accepted by the receiver account. In our scenario, this is the Crypto market-maker.

Accept the cluster placement group resource share and launch EC2 instances
The receiver account can now accept the CPG resource share created above:

1. Open the AWS RAM console and select **Resource shares** under **Shared with me**.

2. The CPG resource share will be listed in a **Pending** state. The owner account ID can be viewed and verified at this point.

3. Select the resource share and then select **Accept resource share**.

Crypto Exchange CPG (⬛⬛⬛⬛⬛⬛⬛⬛)

Details and information relating to this resource share.

[Reject resource share]   [**Accept resource share**]

Summary

Name	Owner	Invitation date	Status
Crypto Exchange CPG	⬛⬛⬛⬛	2022/10/14	🕐 Pending

ARN	Receiver		
arn:aws:ram:eu- west-1: ⬛⬛⬛ :resource -share/ ⬛⬛⬛⬛ ⬛⬛⬛⬛	⬛⬛⬛⬛		

4. The resource share status will change to **Active** in this receiver account. The status under the **Shared principals** section in the owner account will also transition from **Associating** to **Associated**.

5. EC2 instances can now be launched into this shared CPG inside of the receiver account. This can be specified in the Amazon EC2 console on the **Launch instance** page under **Advanced details** using the **Placement group** dropdown menu. The placement group ID allows for the group to be uniquely identified, since the account may have existing groups with identical group names. When launching instances into a shared cluster placement group using the AWS CLI or AWS SDK, you must use the placement group ID to avoid group name conflicts for EC2 instance launches. The placement group will be listed as "**Shared: Yes**" to show that it has been shared with this AWS account:

Placement group Info

Select	▲
🔍	
Select	✓
Crypto Exchange CPG Strategy: cluster [Shared: Yes]	pg- ⬛⬛⬛ 15f7fd178

6. Using the AWS CLI in the owner and receiver AWS accounts, we can execute a describe-instances command, using the placement group ID to *filter instances* successfully launched into the shared CPG. Note the different AZ names which, as mentioned previously, map to the same physical AZ ID in these accounts:

```
Owner AWS Account # aws ec2 describe-instances \
> --filters Name=placement-group-id,Values=pg-███████15f7fd178 \
> --query 'Reservations[*].Instances[*].[InstanceId,Placement]'
[
    [
        [
            "i-██████5675f7a22",
            {
                "AvailabilityZone": "eu-west-1a",
                "GroupName": "Crypto Exchange CPG",
                "Tenancy": "default"
            }
        ]
    ]
]
Owner AWS Account # ▯
```

```
Receiver AWS Account # aws ec2 describe-instances \
> --filters Name=placement-group-id,Values=pg-███████15f7fd178 \
> --query 'Reservations[*].Instances[*].[InstanceId,Placement]'
[
    [
        [
            "i-██████d6fe287cb",
            {
                "AvailabilityZone": "eu-west-1c",
                "GroupName": "Crypto Exchange CPG",
                "Tenancy": "default"
            }
        ]
    ]
]
Receiver AWS Account # ▯
```

Network performance advantages of using Amazon EC2 cluster placement groups

Instances launched into a CPG can benefit from improved network performance across numerous dimensions:

- Lower round trip latencies for network traffic
- Increased packet processing rates
- Increased per-flow throughput limits

Per-flow throughput limits are less relevant to most live market-making use-cases, since these network traffic patterns typically involve relatively small payload sizes. For example, receiving price feeds or executing trades on an exchange involves the transfer of small amounts of data over a given period of time. However, market-makers will regularly perform several offline functions, such as back-testing or asynchronous market data distribution, where increased per-flow throughput limits can reduce the time required to move data on the network between application components. These functions are used to model trading strategies using historical price trends or to provide large amounts of data to analysts for market research purposes. In these scenarios, increased per-flow throughput limits can allow these applications to scale their network utilization by using fewer network connections and application processes.

Network latency and packet processing rates are dimensions that have direct relevance to live trading performance. By making sure of network proximity between participating instances, shared CPGs can provide an advantage to Crypto market-makers by making sure that they experience the lowest network times to a Crypto exchange running within the same AZ. This assists with reducing competitive disadvantages incurred from "hunt engines" created by other market-makers that randomly provision and cull instances in an AZ to discover the lowest latency paths to a given Crypto exchange. In effect, shared CPGs provide a market-maker assurance that their trading engines are placed optimally on the network when compared to other instances in the AZ whose placement has been randomly provisioned outside of a shared CPG.

Achievable latency figures are dependent on numerous factors, including operating system (OS) tuning, application composition of a given trading engine, and the associated business logic. However, purely from a network perspective, the graphs in this section provide an example of latencies and packet processing rates that can be

achieved both inside and outside of a CPG. Tests were conducted on network optimized EC2 c5n instances, using the netperf performance benchmarking tool. Instances were located in separate Amazon VPCs connected via Amazon VPC peering. Note that these graphs don't illustrate the lowest latencies or highest rates achievable on EC2 c5n instance types. These tests were purposely capped and run with default configurations without implementing any further application or OS optimizations. The purpose here is to highlight the advantages provided by using a CPG. Therefore, the actual figures achieved are less important than the relative differences observed between flows inside and outside of a CPG. Performance benefits can also differ depending on the AZ used – size and geographical layout impact latency. We recommend that customers always benchmark workloads to validate real-world figures for their specific applications.

The following graph illustrates latencies for TCP request/response round trip times across multiple flows:
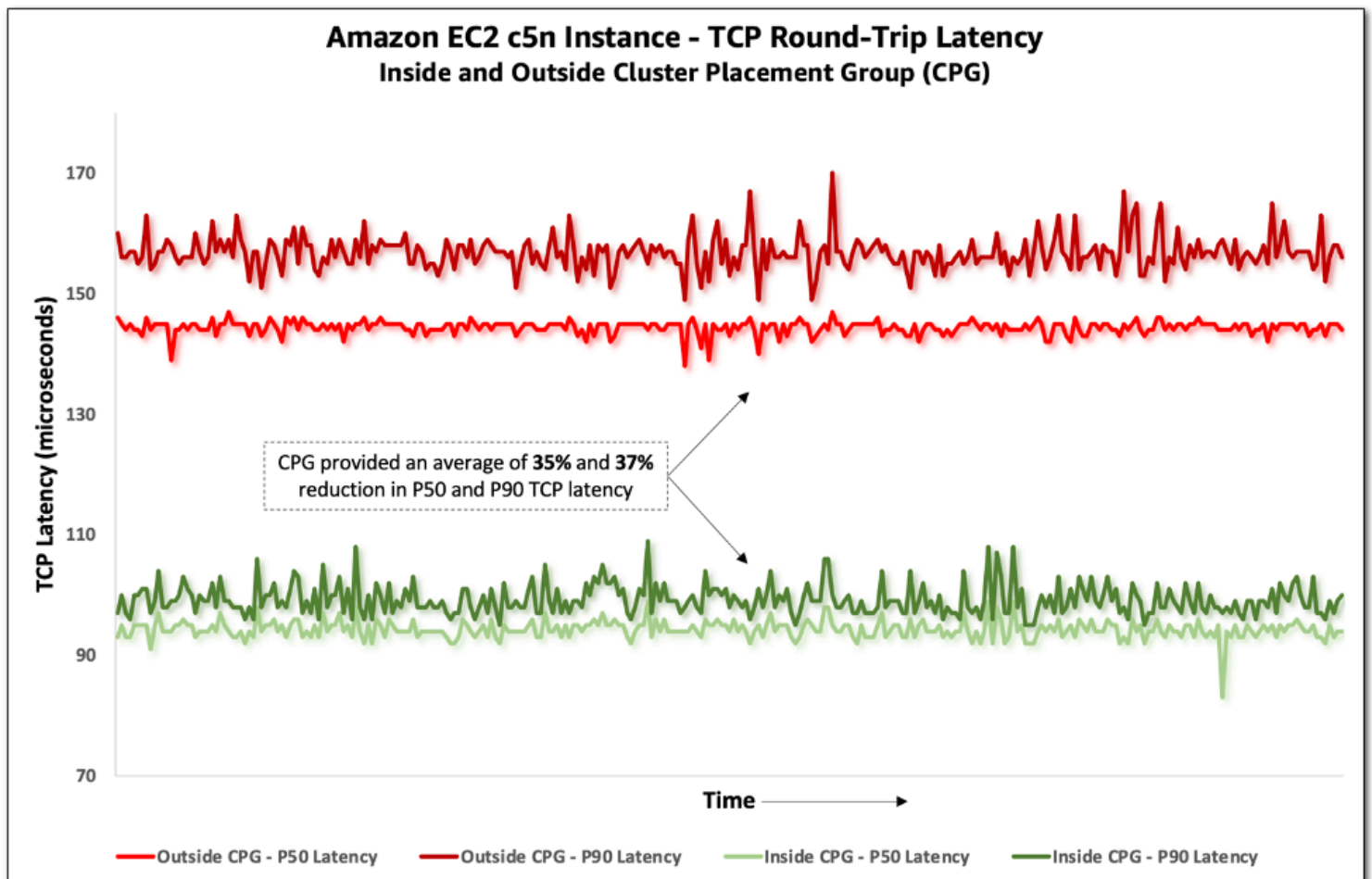
As shown above, the CPG provided a 35% and 37% reduction in P50 and P90 round trip latencies respectively. Market-makers typically monitor time on the network in microseconds. Therefore, sub-millisecond improvements represent a business impacting optimization opportunity. Although price feeds from Crypto exchanges are often consumed via TCP Websocket connections, UDP is frequently used for market data distribution to remove TCP overheads, inherent in the protocol, and reduce network latency. The following graph illustrates a similar advantage provided by CPGs for UDP flows:
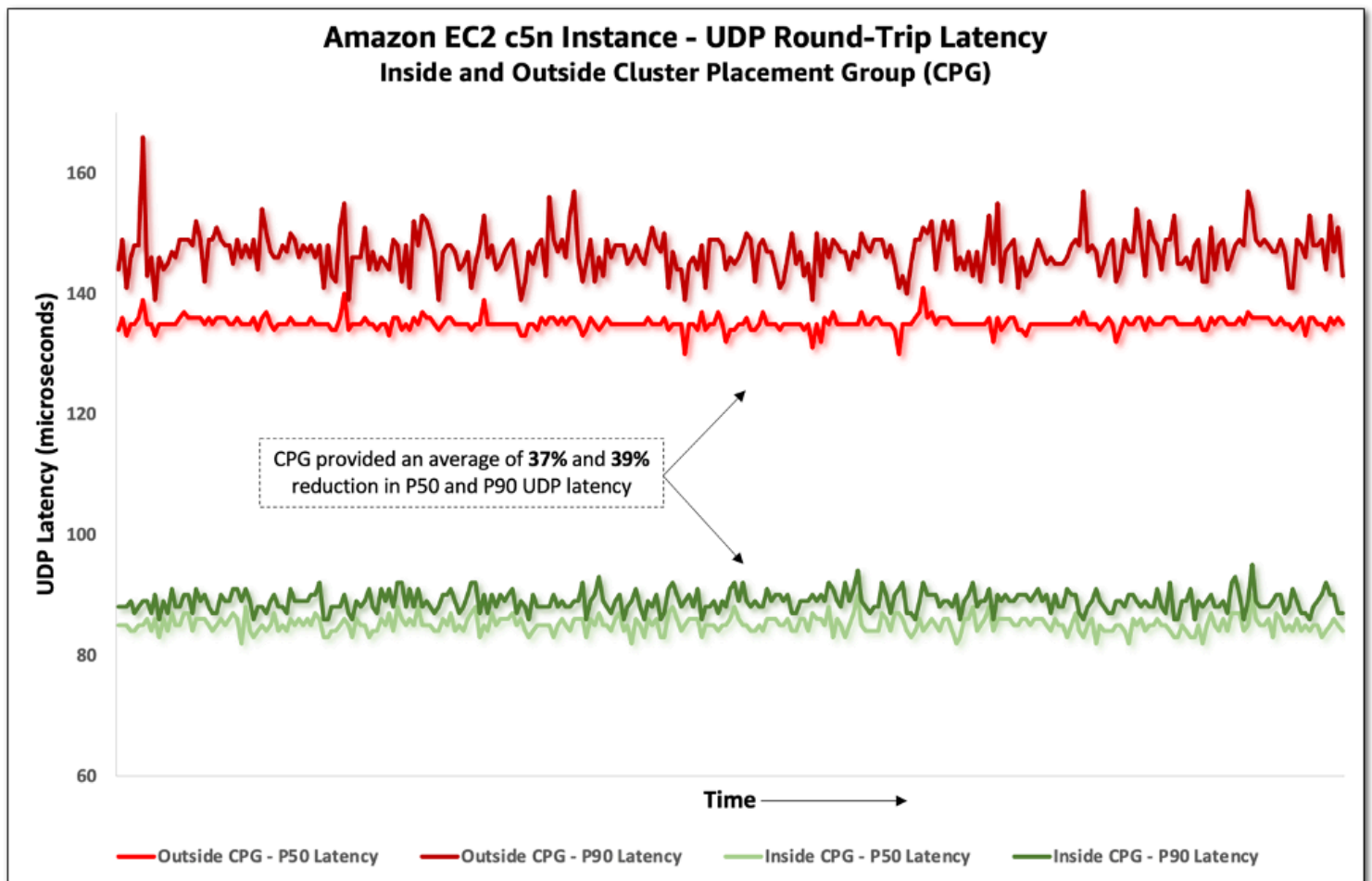
These graphs also demonstrate a smaller variance between P50 and P90 latency for instances inside of the CPG, thereby reducing the range of network latency incurred. This allows customers to be more deterministic about the nature of network performance that they will experience. Lower network latencies also generally improve packet processing rates. The following graph illustrates this using the same test configuration as shown previously with many concurrent flows:

Amazon EC2 c5n Instance - Packet Processing Rates
Inside and Outside Cluster Placement Group (CPG)
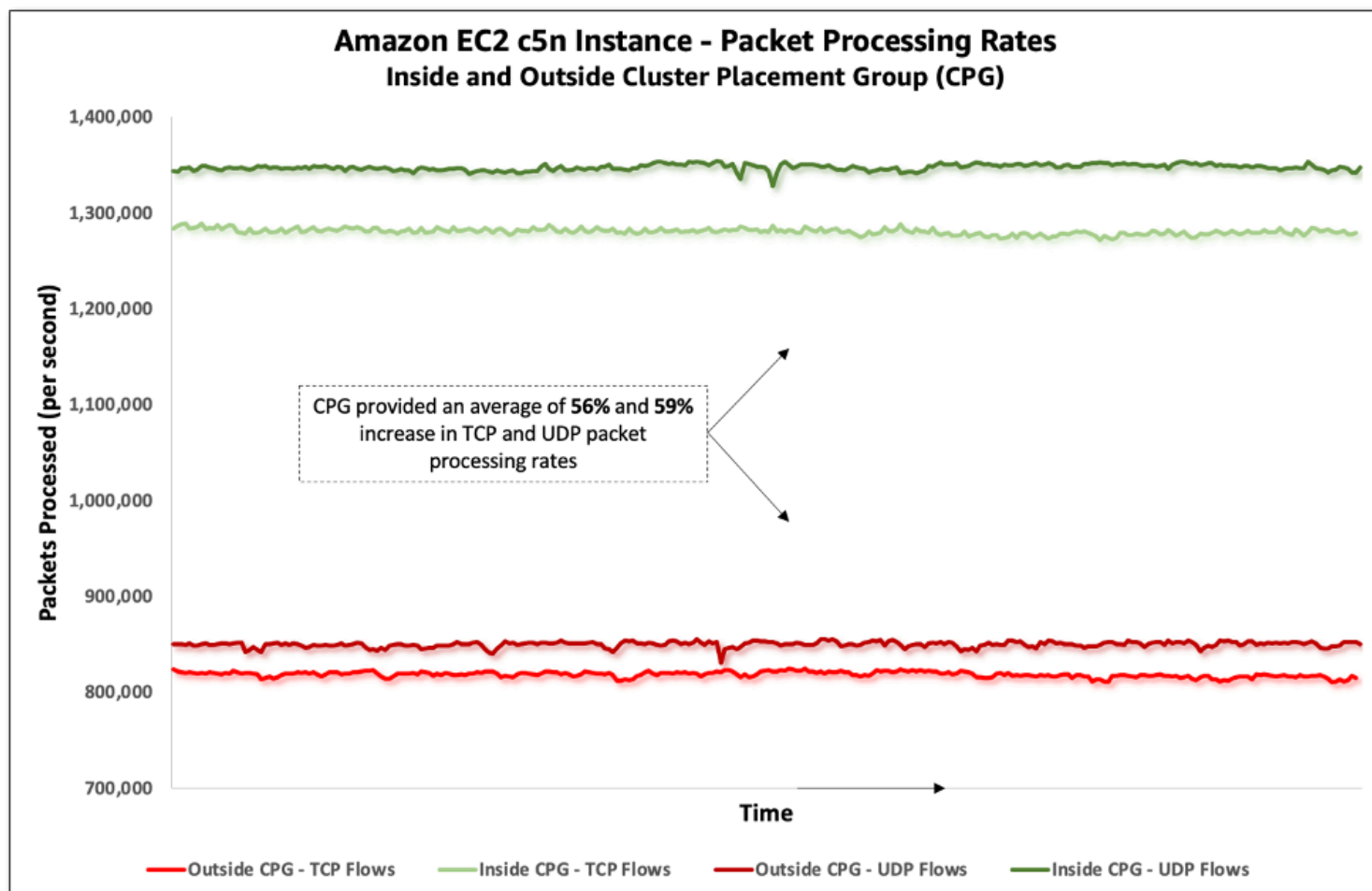
Packet processing rates increased significantly for instances running inside the CPG – by 56% and 59% for TCP and UDP flows respectively. EC2 instance packet processing rates are important to market-makers, since an increase can improve profitability in a number of scenarios – for example during times of market volatility when there is a need to process rapidly changing tick data and to execute a larger number of trades over specific time periods.

For latency sensitive workloads, we recommend that customers choose EC2 network optimized instances, such as the c7gn, c6gn, c6in, m6in, r6in, c5n, m5n or r5n families, and enable Amazon EC2 enhanced networking. Customers should also consider using .metal variants, wherever offered within these instance families, which provide direct "bare metal" access to the underlying host, and further optimizing latencies in the instance networking stack. Furthermore, we recommend that customers benchmark their applications across various EC2 instance types to identify the best choice for their specific workload profiles. As AWS continues to develop the AWS Nitro System, new instance families can provide further optimizations in offloading network operations to the AWS Nitro hypervisor. Finally, although a detailed analysis is beyond the scope of this post, customers can also benefit from using the Intel Data Plane Developer Kit (DPDK) to further increase packet processing performance by integrating DPDK code into their latency sensitive application traffic flows.

Logical network architecture considerations

We've discussed how CPGs make sure of physical network proximity between EC2 instances in the same AZ. However, beyond this physical foundation, customers must establish and allow logical network connectivity between their instances. For instances within the same VPC, this is easily achieved by updating security groups. However, in our

example scenario of a Crypto exchange and Crypto market-maker operating out of different AWS accounts, and therefore VPCs, there is a need to configure additional inter-VPC networking connectivity.

AWS provides numerous fully-managed services to achieve logical connectivity across VPCs and AWS accounts, including AWS Transit Gateway and AWS PrivateLink. Both of these services enable customers to logically connect VPCs at scale. These services are built on AWS Hyperplane – a Network Function Virtualization platform that provides transparent routing and switching capabilities at an extremely large scale (thousands of VPCs). Although this level of detail is typically not relevant to customers, in the context of CPGs this information becomes significant. Hyperplane was developed to allow customers to scale their logical connectivity on AWS. To achieve this, the platform creates logical objects (Gateways and Endpoints) that represent additional hops on the network. As well as marginally increasing hop count, Hyperplane components aren't situated in the physical scope of a CPG. Consequently, placing Hyperplane-based connectivity services in the path between members of a CPG can counteract the benefits gained by optimizing proximity between those EC2 instances.

Therefore, for use-cases that require the use of shared CPGs across VPCs, such as market-making, we recommend the use of Amazon VPC peering to achieve logical connectivity between Crypto exchange and market-maker VPCs. This minimizes the number of physical and logical network hops and provides the lowest levels of network latency. The following diagram illustrates an example logical and physical view when using VPC peering with shared CPGs:
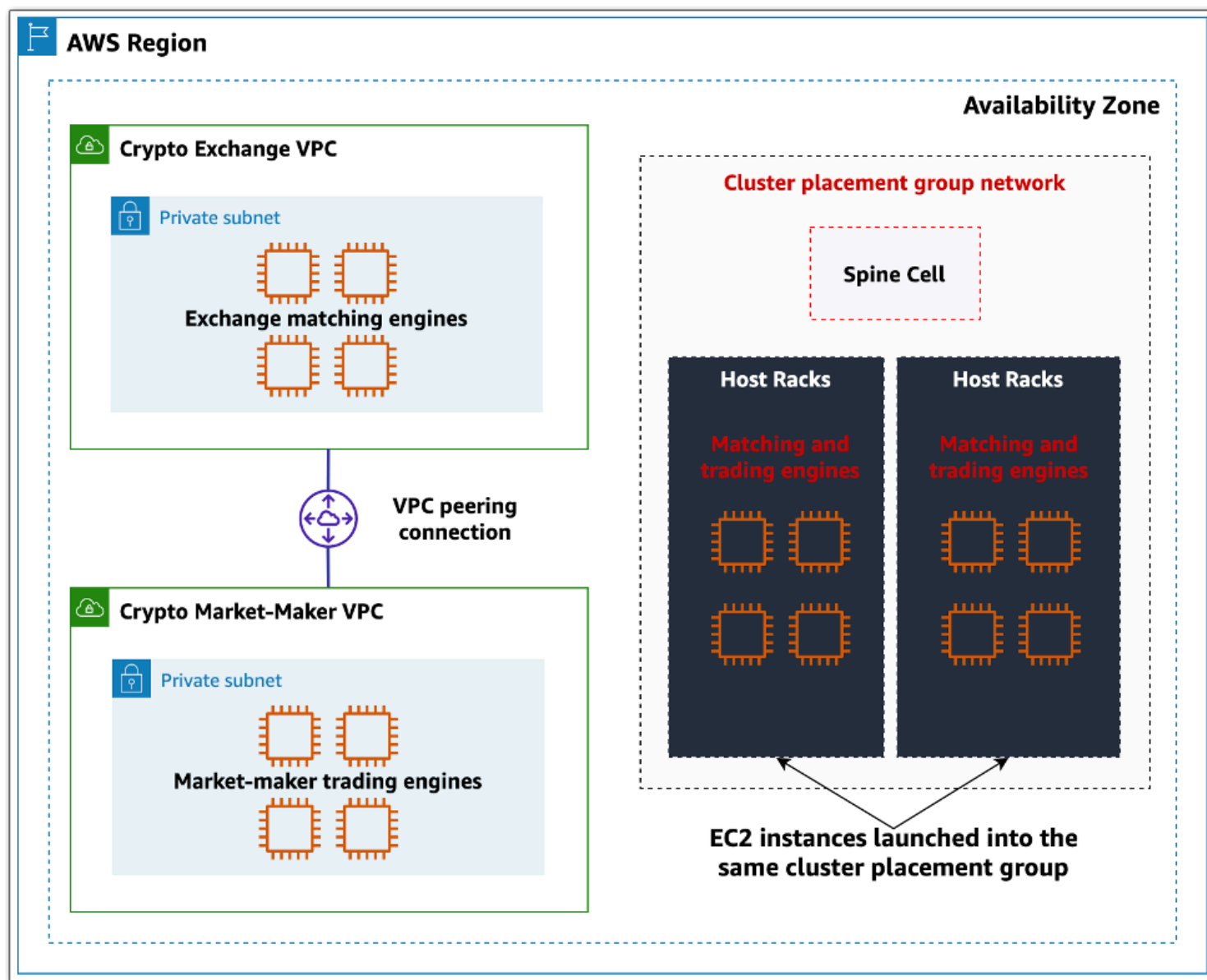
Traffic carried between an intra-region VPC peering connection incurs no additional network hops or latency when compared to traffic flowing within a single VPC. Within the same AZ, this traffic also incurs no additional cost for network data transfer. Since VPC peering doesn't support overlapping CIDR ranges, coordination is required to manage and use appropriate private IP space for VPC entities on either side of the VPC peering connection. Where VPCs reside in separate AWS accounts or Organizations, creating peering connections requires IAM permissions that cross these account or Organization boundaries to modify and update VPC network configurations. Although this is possible with IAM, security conscious customers may be hesitant to allow third-party access to modify something as sensitive as network connectivity configuration.

With this in mind, we've published a GitHub repository that automates this process. Using custom AWS CloudFormation resources, the code provides a self-service approach for a provider (e.g., Crypto exchange) to pre-allocate VPC CIDRs, and it allows for the acceptance of VPC peering connections, as well as route table and security group updates, without the need to provision IAM permissions to third-parties.

Conclusion

In this post, we discussed the real-world use case of Crypto market-making to illustrate how exchanges can use the newly-released Amazon EC2 shared CPG feature to share placement groups with market-makers in other AWS accounts. This provides market-makers with the opportunity to locate their trading engines closer to Crypto exchanges on the AWS Cloud, thereby improving network latency and packet processing rates for a competitive advantage and better profitability. We also illustrated the magnitude of performance optimization that can be achieved using CPGs containing one type of EC2 instance running a specific set of network testing benchmarks. Finally, we addressed logical inter-VPC connectivity and explained why VPC peering is the recommended option in the context of CPGs spanning multiple VPCs. We also shared a GitHub repository that provides self-service automation for securely creating Amazon VPC peering connections at scale between Organizations.

TAGS: Financial Services