JAPAN | 2024

aws SUMMIT

AWS-01

Dive deep on Amazon S3

焼尾 徹
アマゾン ウェブ サービス ジャパン合同会社
シニアストレージスペシャリストソリューションアーキテクト

There is no compression algorithm for experience

経験を積むための圧縮アルゴリズムは存在しない

目次

Amazon S3 の ３つの歩み

リアクティブ　　脅威モデリング　　プロアクティブ

Amazon S3 とは?

350 以上のマイクロサービス
すべての AWS リージョンで

クライアント

フロントエンド

Web サーバ, DNS, ネットワーク

インデックス

ストレージ層への Key/Value マップ

ストレージ層

媒体でのデータの冗長性確保

クライアント

フロントエンド
Web サーバ, DNS, ネットワーク

インデックス
ストレージ層への Key/Value マップ

ストレージ層
媒体でのデータの冗長性確保

- フロントエンドの理解
- インデックスの理解
- ストレージ層の理解
- Amazon S3 Express One Zone
- 誤削除への対策

ピークネットワークは1PBの移行を超える

スケールするように実装するのが大切

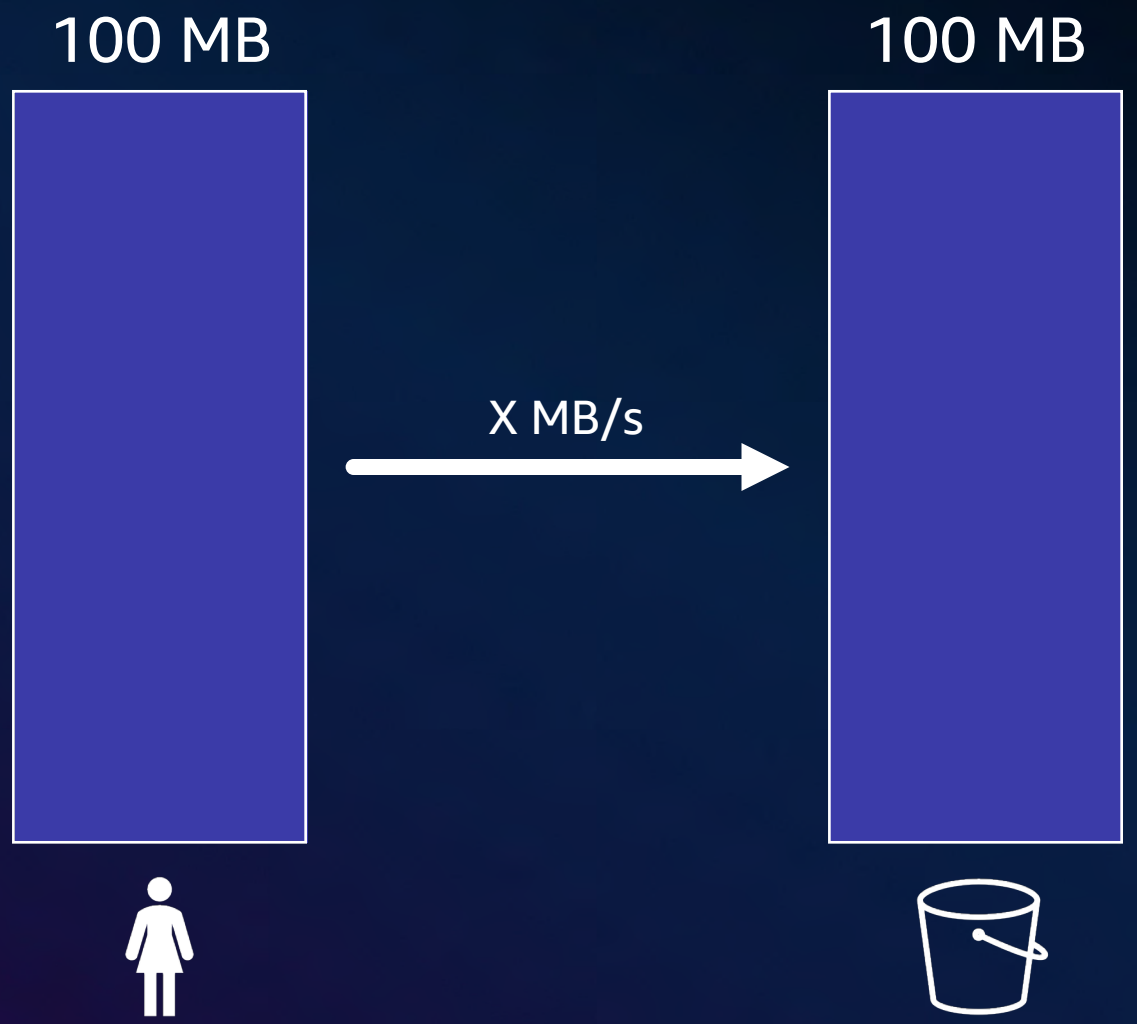
緩和策として対応できる仕組み

並列で PUT 処理できるように、マルチパートアップロードを利用する

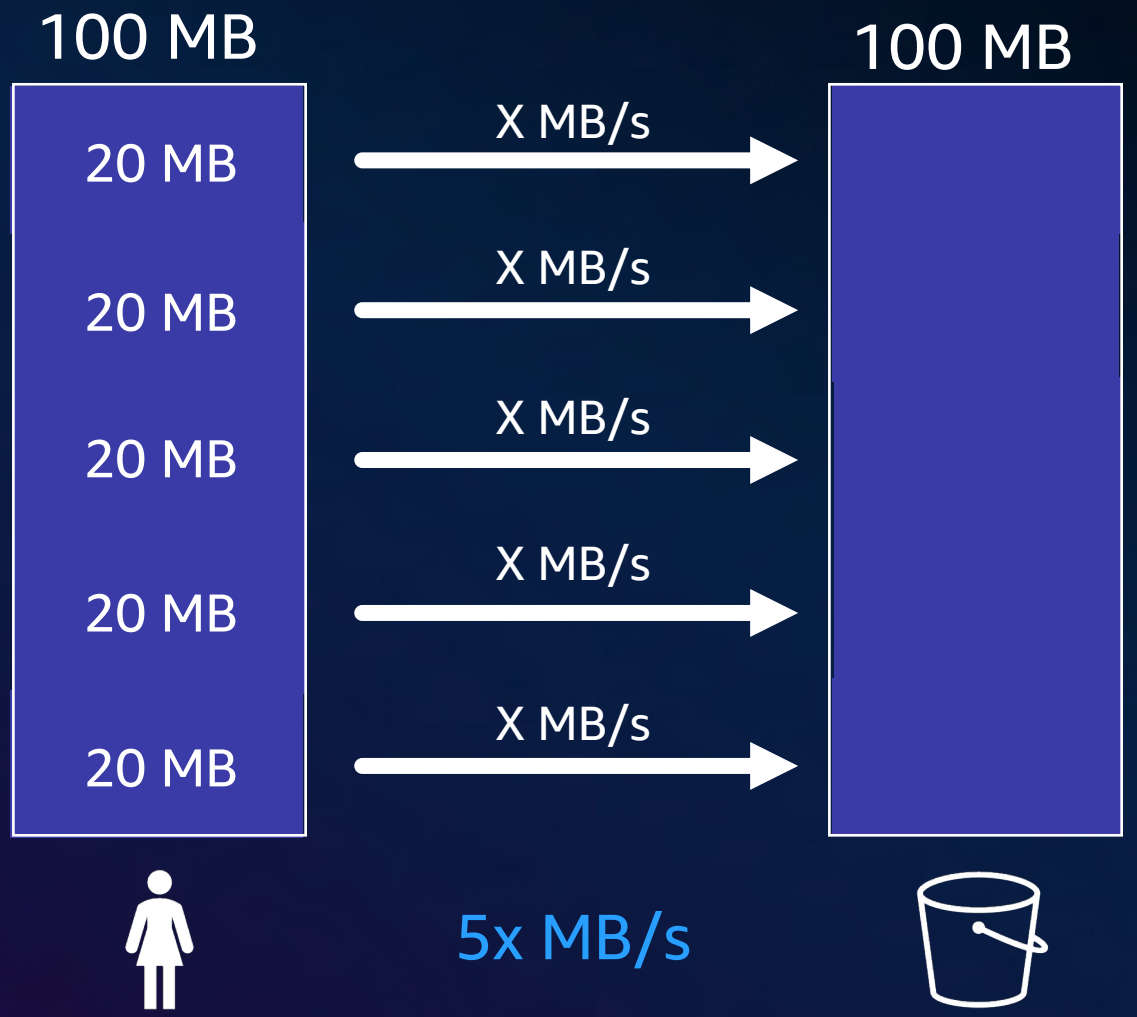
並列で GET 処理できるように、レンジを利用する

リクエストを複数の IP へ分散する

マルチパートアップロードとレンジ



分割したパートを並列でデータ転送する



分割したパートを並列でデータ転送する

マルチパートアップロード API の活用

```
create-multipart-upload
  [--acl <value>]
  [--grant-full-control <value>]
  [--grant-read <value>]
  [--grant-read-acp <value>]
  [--grant-write-acp <value>]
  [--metadata <value>]
  [--storage-class <value>]
  [--server-side-encryption <value>]
  [--ssekms-key-id <value>]
  [--bucket-key-enabled]
  [--tagging <value>]
  [--object-lock-mode <value>]

--checksum-algorithm <value>
--key <value>
--bucket <value>
```

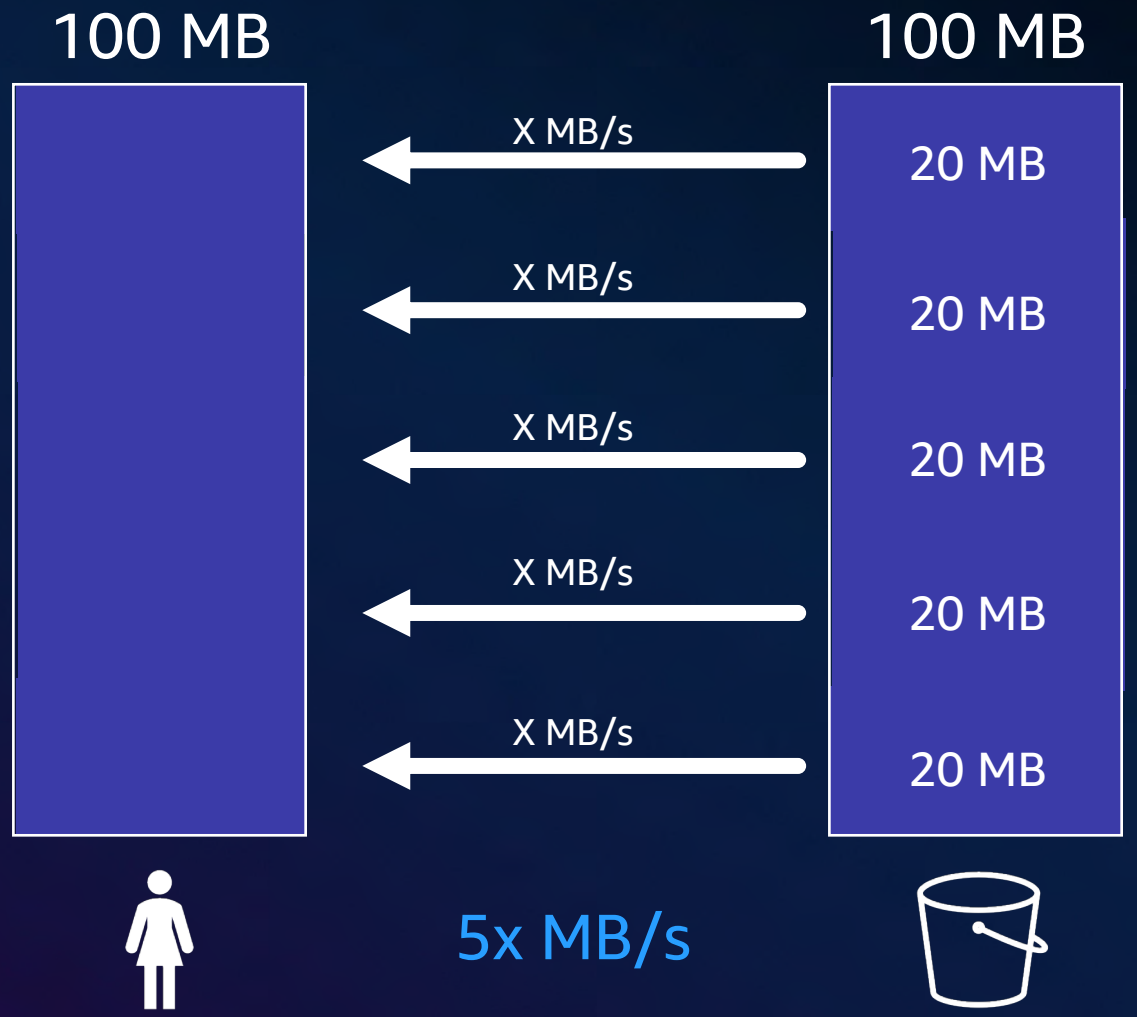
```
upload-part
--part-number <value>
--upload-id <value>
--key <value>
--bucket <value>
  . . . . . .
upload-part
--part-number <value>
--upload-id <value>
--key <value>
--bucket <value>
```

n 回行う

```
complete-multipart-upload
--upload-id <value>
--key <value>
--bucket <value>
```

S3API CLI docs:

データ取得でも同様の並列化が可能

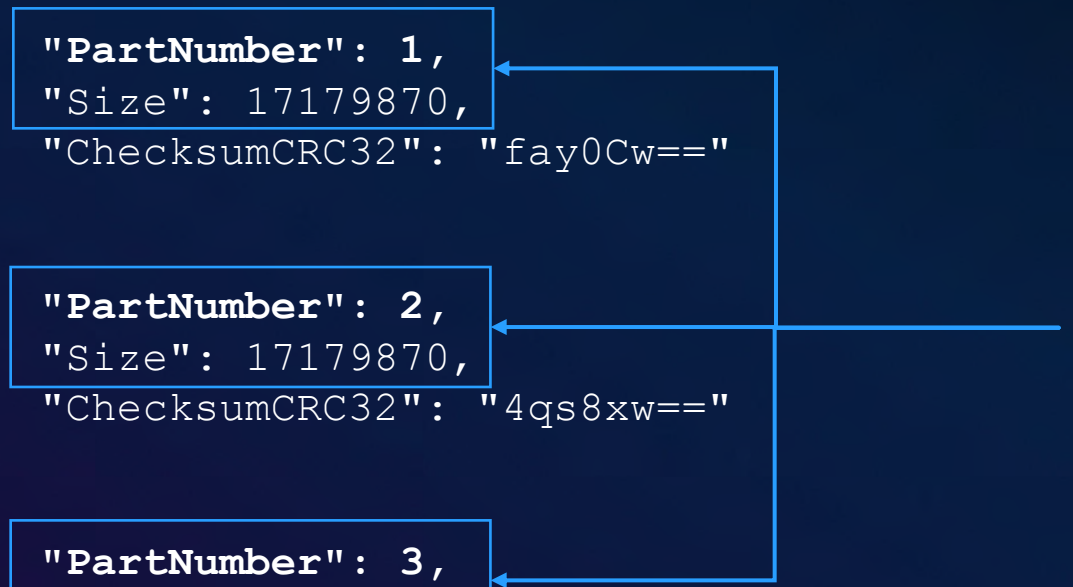


データ取得でも同様の並列化が可能

GET-OBJECT-ATTRIBUTES または HEAD-OBJECT を利用し範囲を取得する

get-object-attributes

```
…
"Parts": [
            {
                "PartNumber": 1,
                "Size": 17179870,
                "ChecksumCRC32": "fay0Cw=="
            },
            {
                "PartNumber": 2,
                "Size": 17179870,
                "ChecksumCRC32": "4qs8xw=="
            },
            {
                "PartNumber": 3,
                "Size": 14692009,
                "ChecksumCRC32": "Ajz83g=="
            }
        ]
…
```



オフセットの計算
もしくは
パート番号の活用

S3API CLI docs:

データ取得でも同様の並列化が可能

レンジゲットリクエストの利用

```
get-object
  [--range <value>]
  [--part-number <value>]

--key <value>
--bucket <value>
<OUTFILE>
```

n 回行う

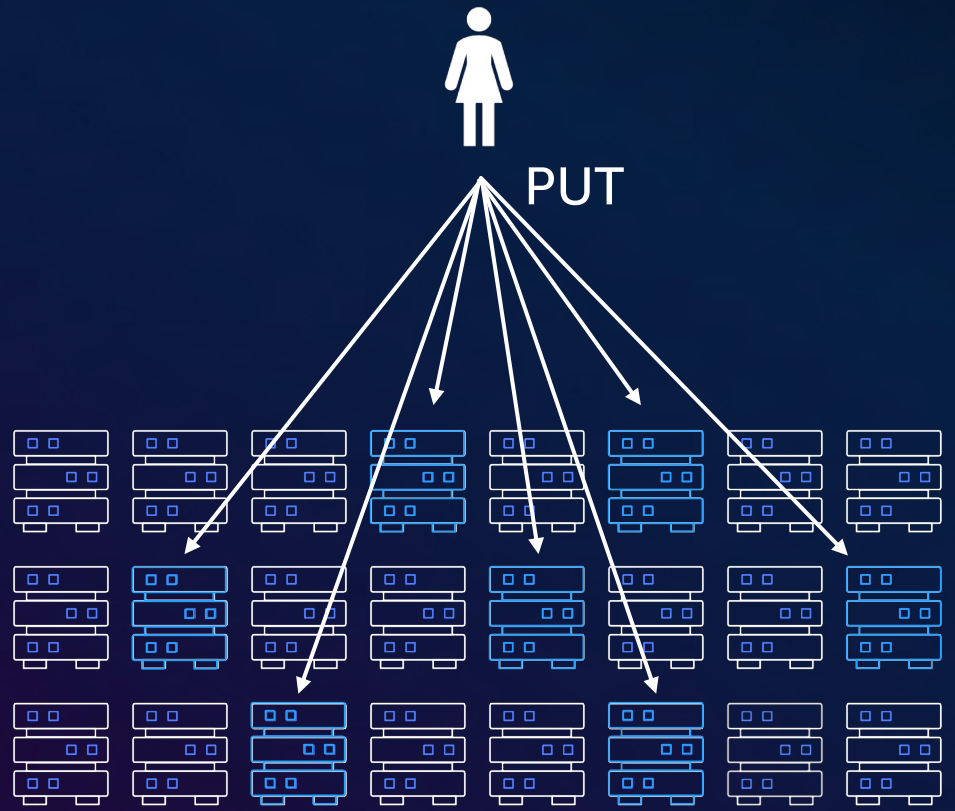
… 繋ぎ合わせて、ストレージに書き込む、というコードを記述する

Range GET RFC:

S3API CLI docs:

フリートにわたってリクエストを分散させる

常に DNS を利用しましょう



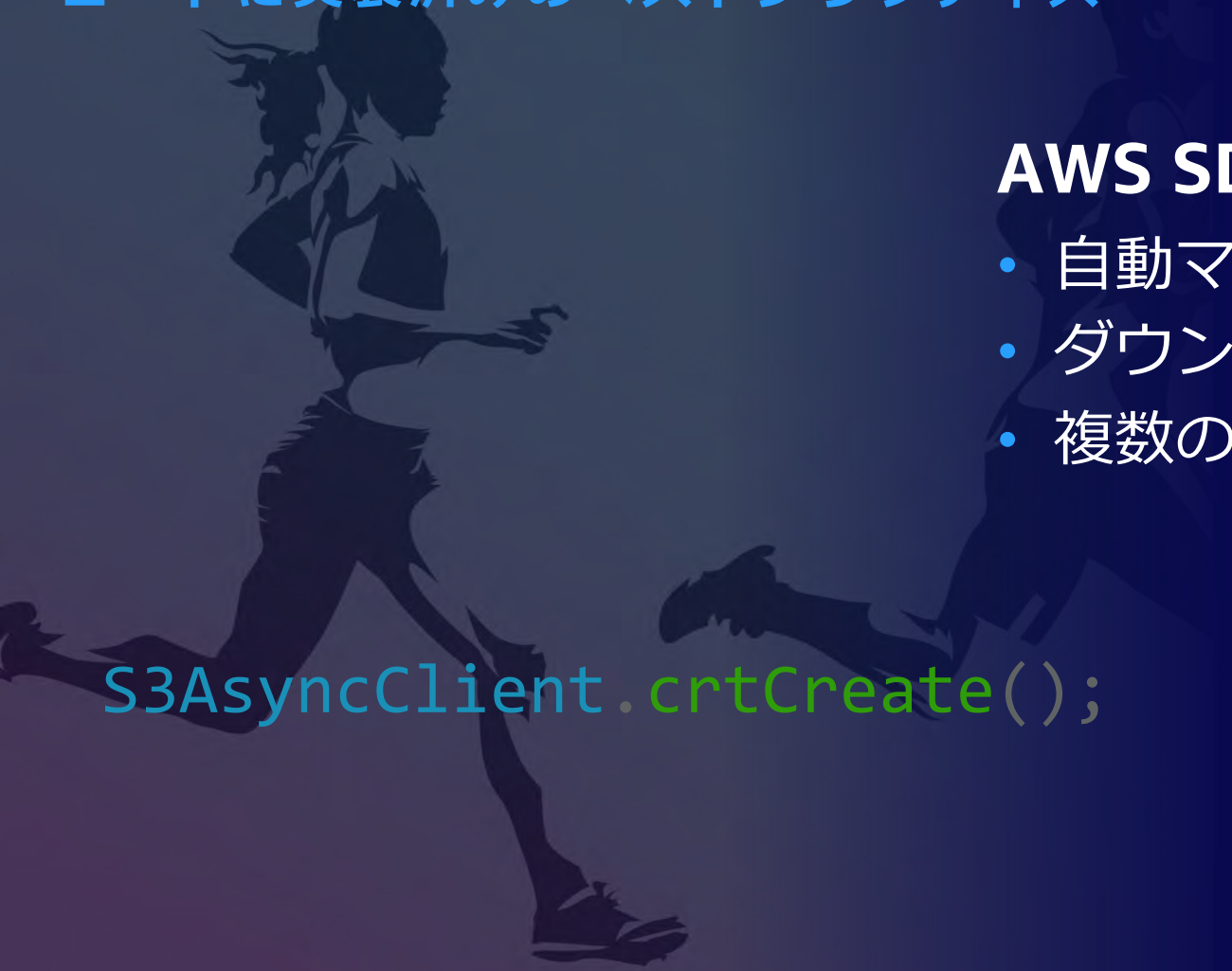

```
# nslookup s3.amazonaws.com

   Server:  192.0.2.3
   Address: 192.0.2.3#53

   Non-authoritative answer:

   Name: s3.amazonaws.com
   Address: 192.0.2.1

   Name: s3.amazonaws.com
   Address: 192.0.2.4

   Name: s3.amazonaws.com
   Address: 192.0.2.5

   Name: s3.amazonaws.com
   Address: 192.0.2.9

   Name: s3.amazonaws.com
   Address: 192.0.2.2

   Name: s3.amazonaws.com
   Address: 192.0.2.6

   Name: s3.amazonaws.com
   Address: 192.0.2.8

   Name: s3.amazonaws.com
   Address: 192.0.2.7
```

複数値回答 (MVA)
Multi-value answer

NEW

Common Runtime (CRT)

コードに実装済みのベストプラクティス



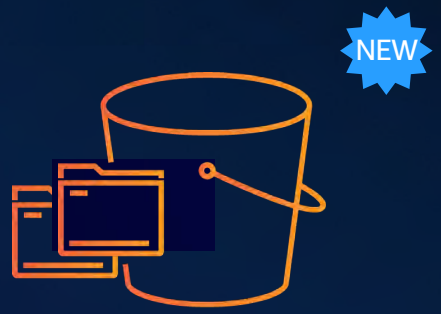
```
S3AsyncClient.crtCreate();
```

AWS SDK のオープンソースコンポーネント

- 自動マルチパートアップロード
- ダウンロード用のレンジ GET の並列化
- 複数の IP を使用する組み込みリトライロジック

CRT info:

Mountpoint



Mountpoint の
ローカルキャッシュ

Mountpoint の
コンテナ対応

Mountpoint の
S3 Express 対応

Amazon S3 ファミリーのマウントポイント

クライアント

フロントエンド
Web サーバ, DNS, ネットワーク

インデックス
ストレージ層への Key/Value マップ

ストレージ層
媒体でのデータの冗長性確保

フロントエンドの理解

インデックスの理解

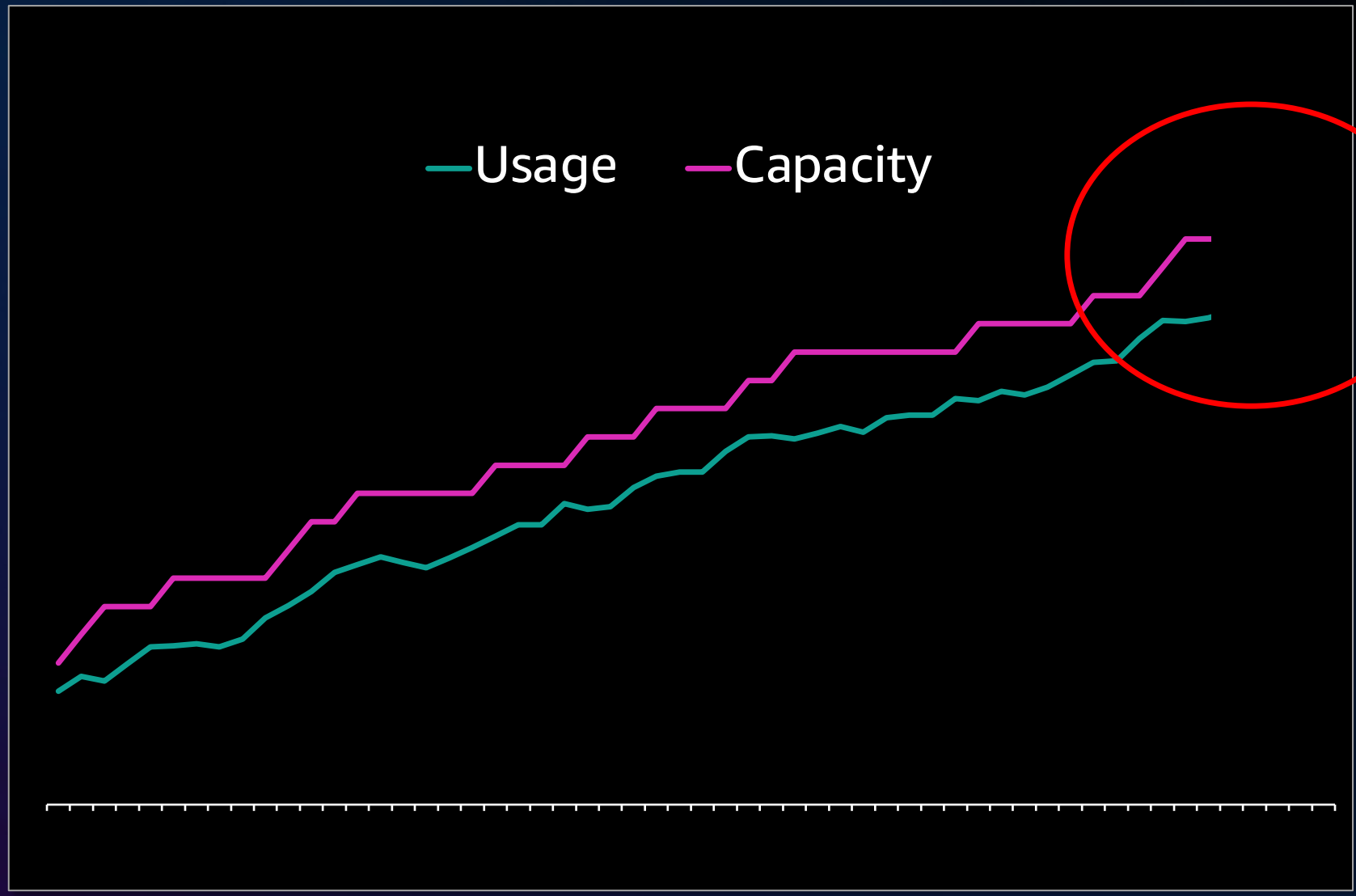
ストレージ層の理解

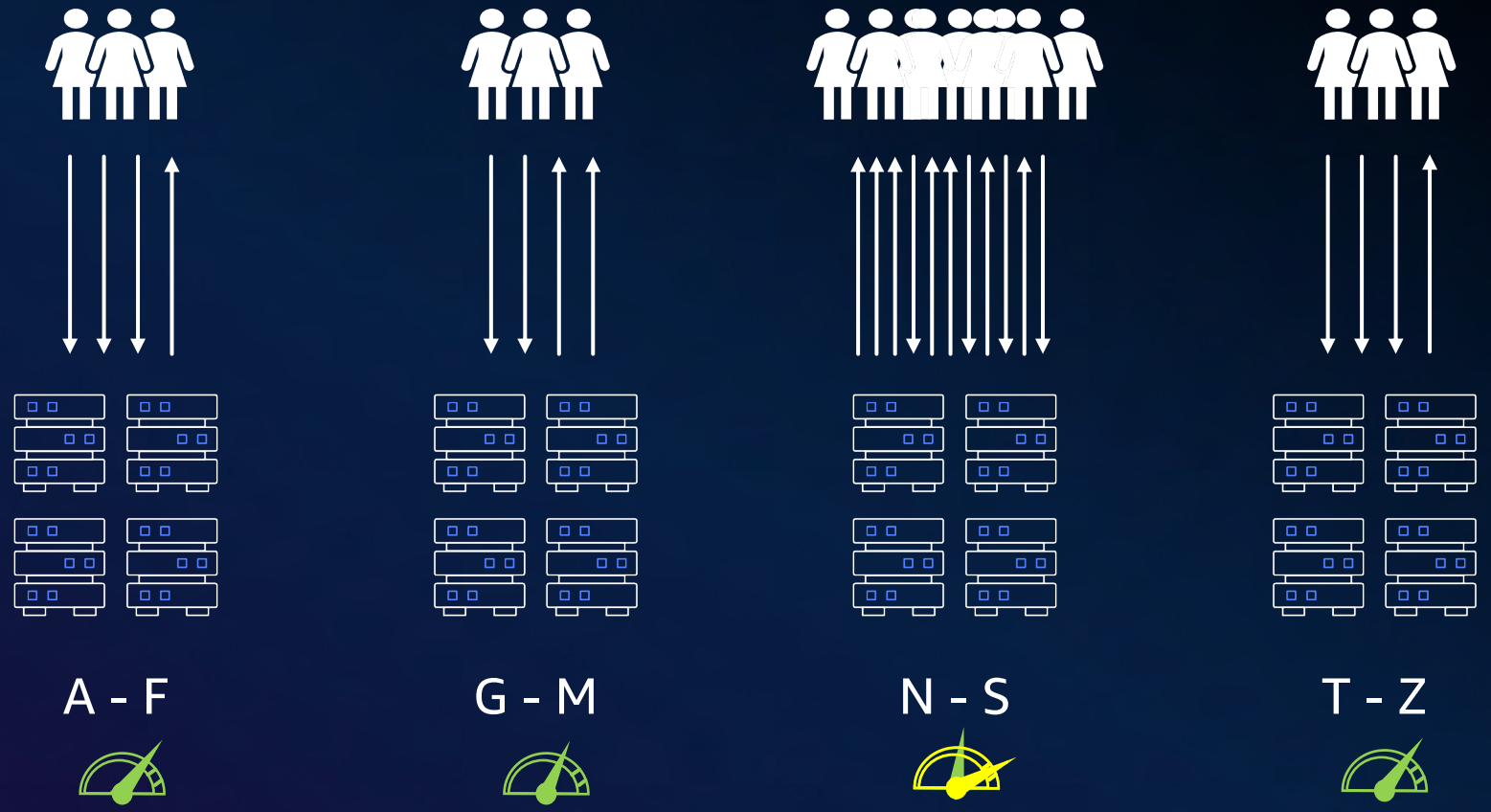
Amazon S3 Express One Zone

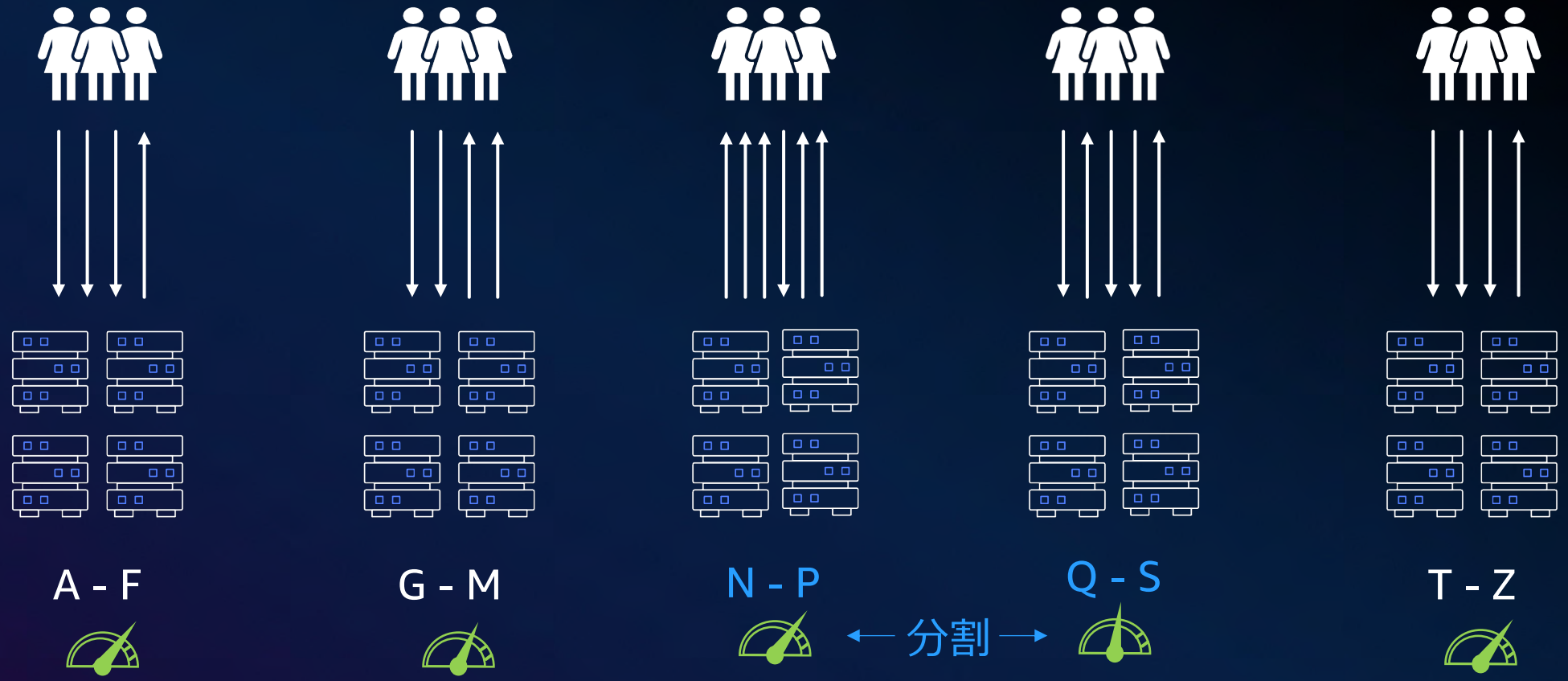
誤削除への対策

S3 への兆のリクエストの理解

秒間 1 億件以上のリクエスト







プレフィックスとは何か?

バケット名の後につづく
任意の文字列

プレフィックスとは何か？

sample-bucket/p
sample-bucket/prefix
sample-bucket/prefix1/
sample-bucket/prefix2/
sample-bucket/prefix1/data/other
sample-bucket/prefix1/data/otherstuff

3,500 PUT リクエスト/秒
5,500 GET リクエスト/秒
プレフィックスあたり

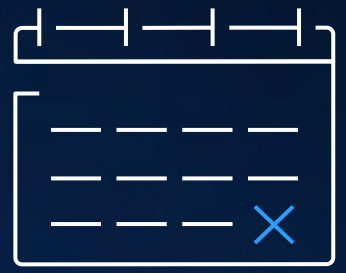

503

sample-bucket におけるトータルのトランザクション/秒 (TPS) は
5,500 x 4 = 22,000 TPS

適切なキーの命名による TPS の最大化

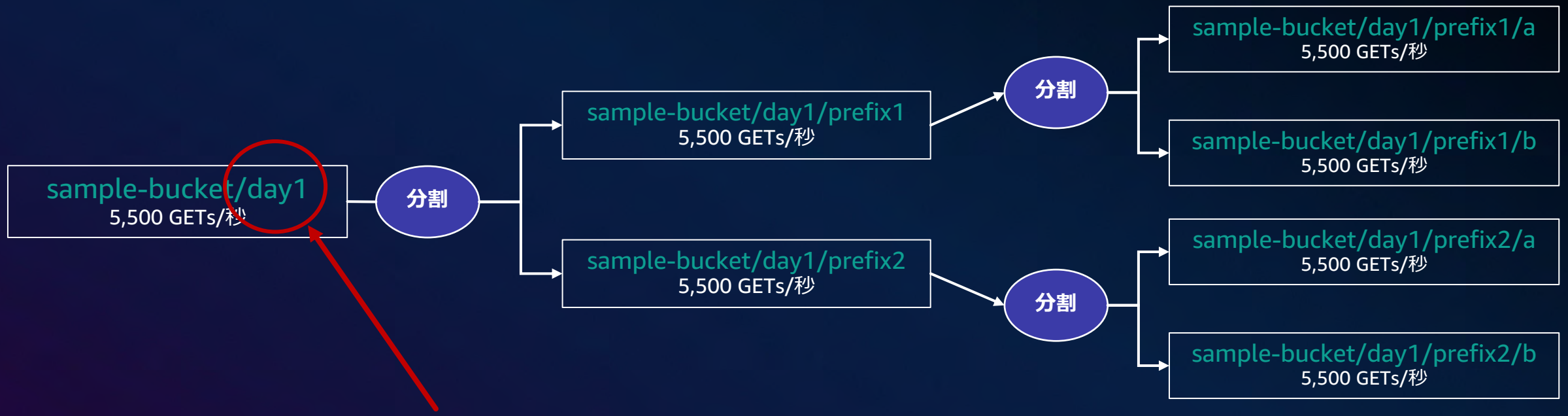
緩和策として対応できる仕組み

キー名のカーディナリティは左に寄せておく



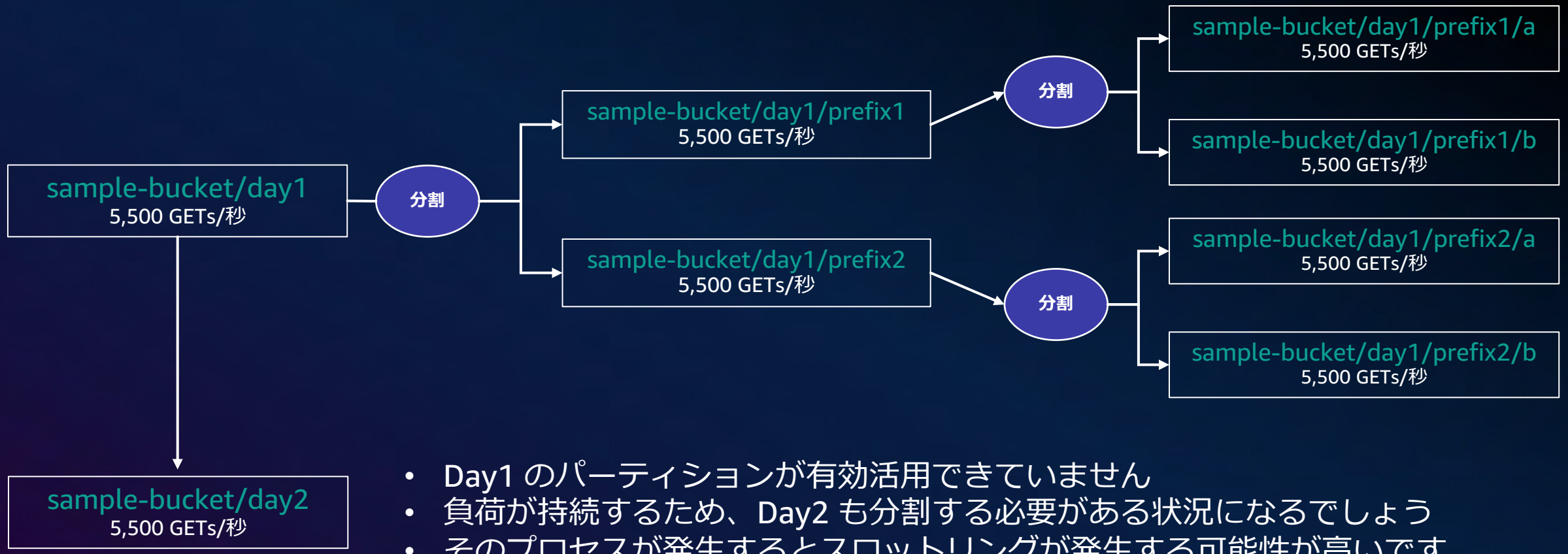
キー名に日付を入れるならばなるべく右側に

キー命名の例（望ましくない例）



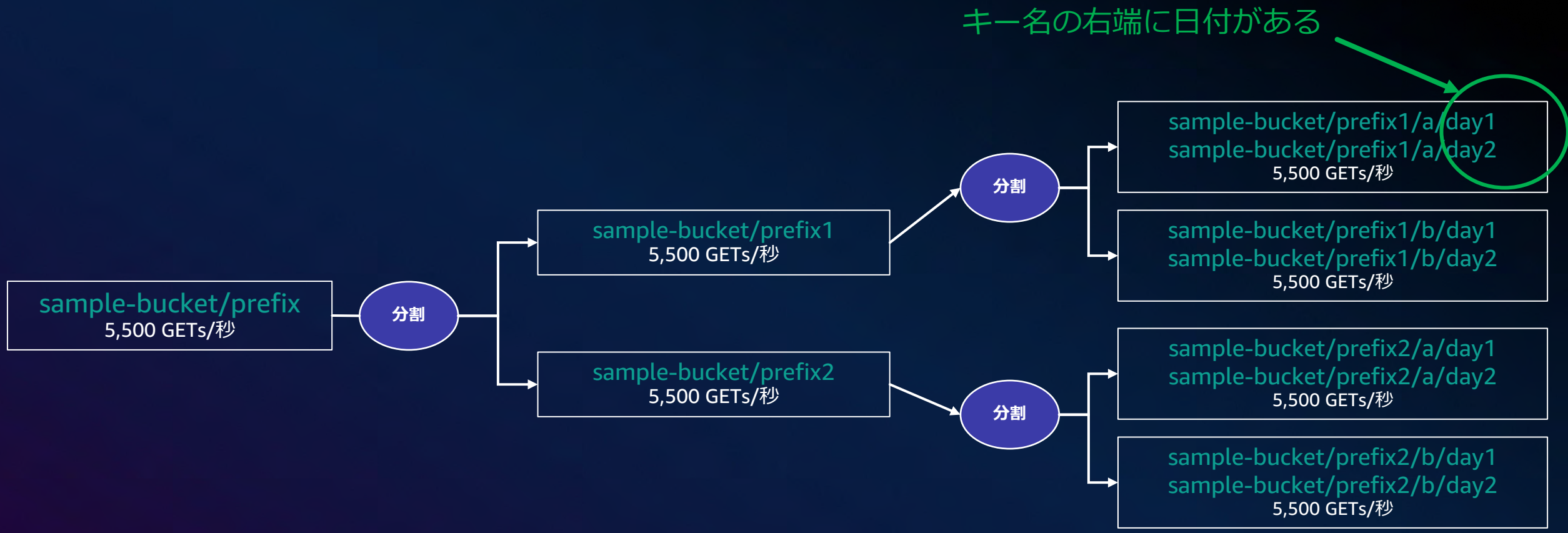
sample-bucket/day1 におけるトータルの TPS は
5,500 x 4 = 22,000 TPS

キー命名の例（望ましくない例）



- Day1 のパーティションが有効活用できていません
- 負荷が持続するため、Day2 も分割する必要がある状況になるでしょう
- そのプロセスが発生するとスロットリングが発生する可能性が高いです

キー命名の例



クライアント

フロントエンド
Web サーバ, DNS, ネットワーク

インデックス
ストレージ層への Key/value マップ

ストレージ層
媒体でのデータの冗長性確保

フロントエンドの理解

インデックスの理解

ストレージ層の理解

Amazon S3 Express One Zone

誤削除への対策

何百万ものバー層の理解

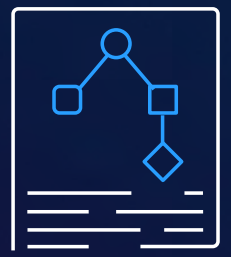
エキサバイトのデータ

99.999999999 %
データ耐久性

デバイス障害に対して

ドライブの故障またはビット列の破壊があった場合でも、
そのドライブに存在するデータを保護する必要があります

どのようにして 11 9s の耐久性を実現するのか



エンドツーエンドでの
リクエストの整合性チェック

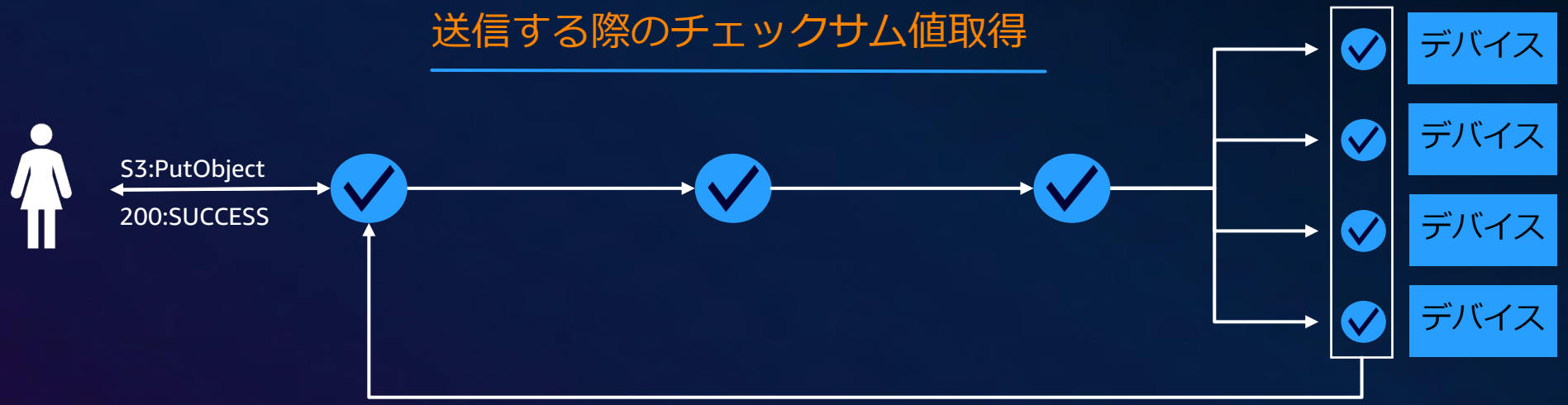
データは常に冗長性の
保たれたデバイスに格納される

保存されたデータの
定期的な耐久性監査

リクエストの整合性チェック

データは冗長性を保って格納され、
ストレージ上でチェックサムを取得します

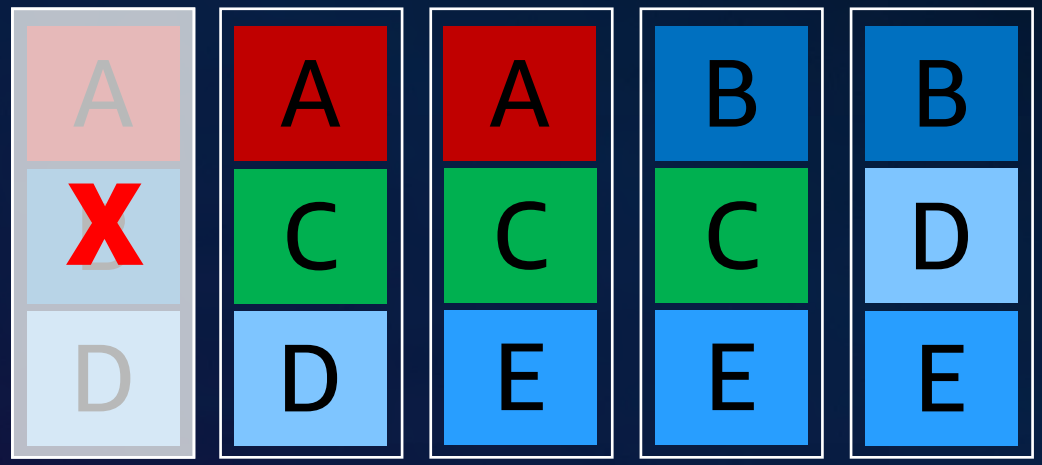
送信する際のチェックサム値取得



保存されたデータとアップロードされたデータの比較

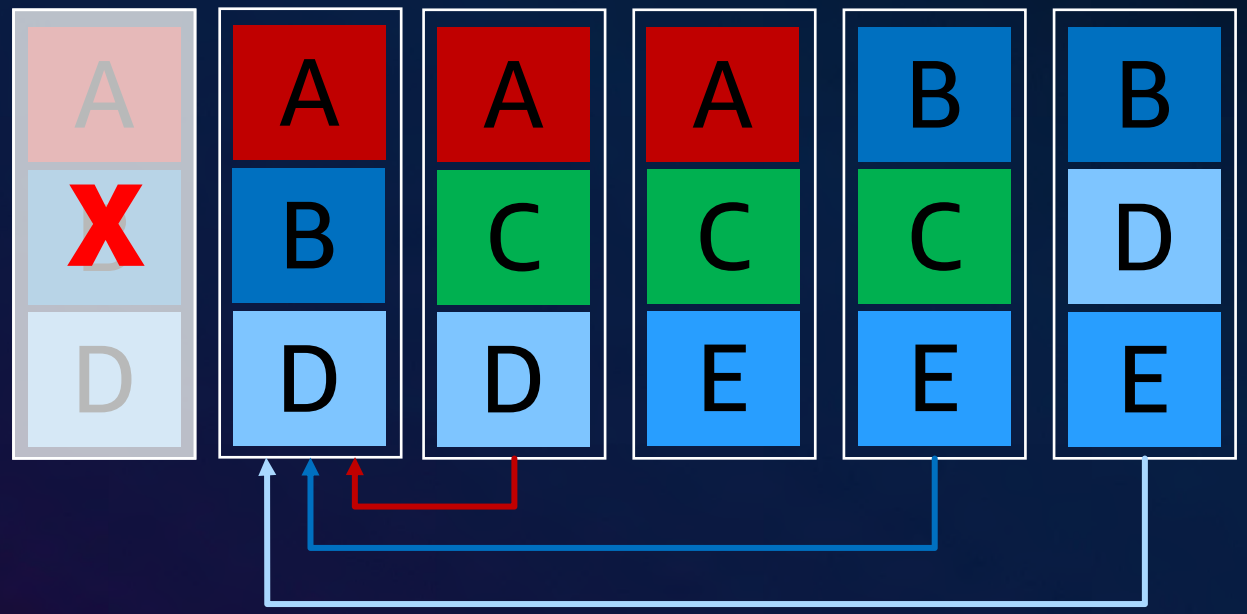
データは常に冗長化されたデバイスに格納される

5 台のデバイス、それぞれデータは異なる

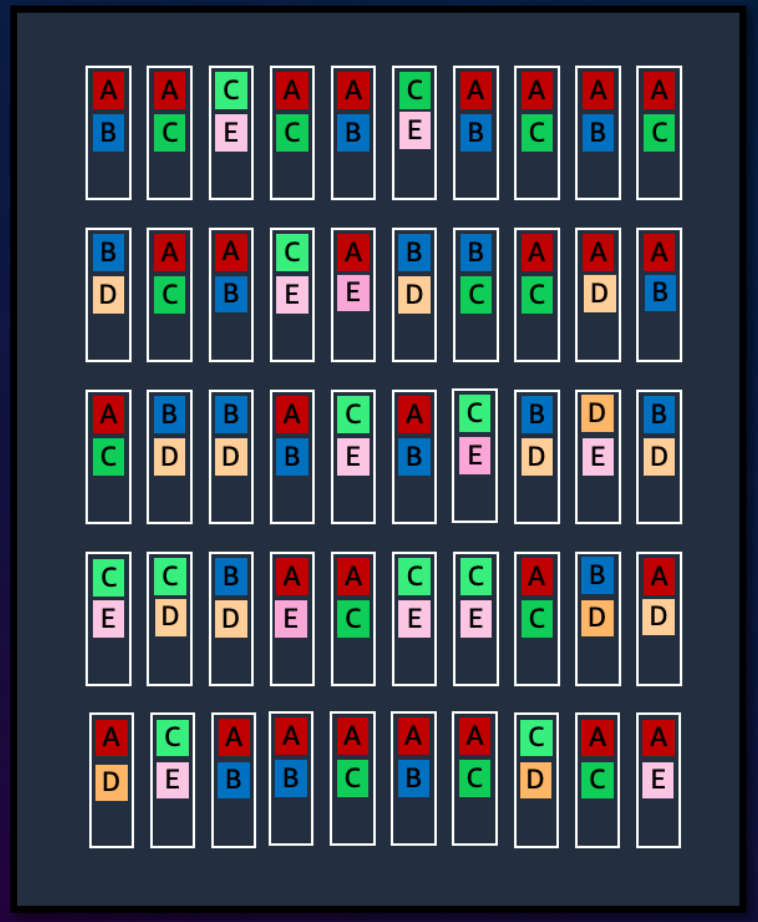


データは常に冗長化されたデバイスに格納される

5 台のデバイス、それぞれデータは異なる



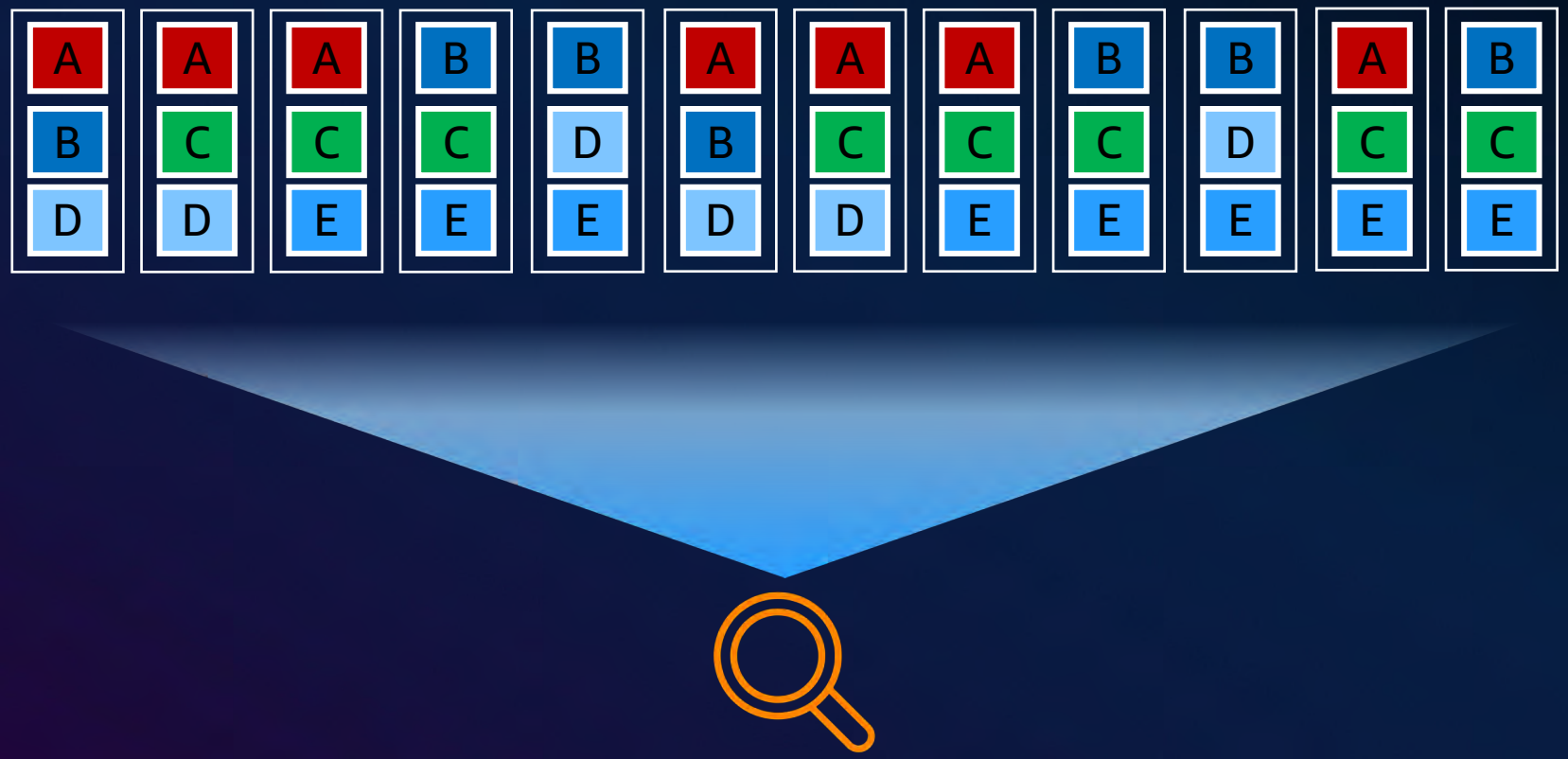
データは常に冗長化されたデバイスに格納される



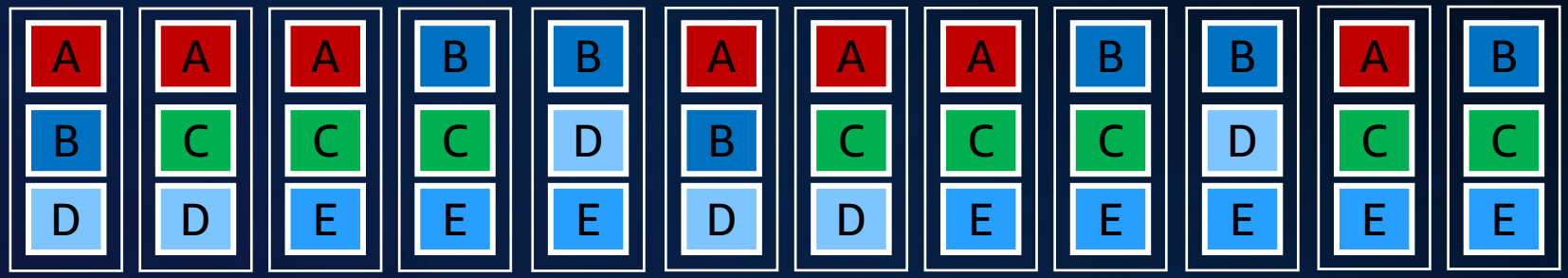
大規模に実現しています

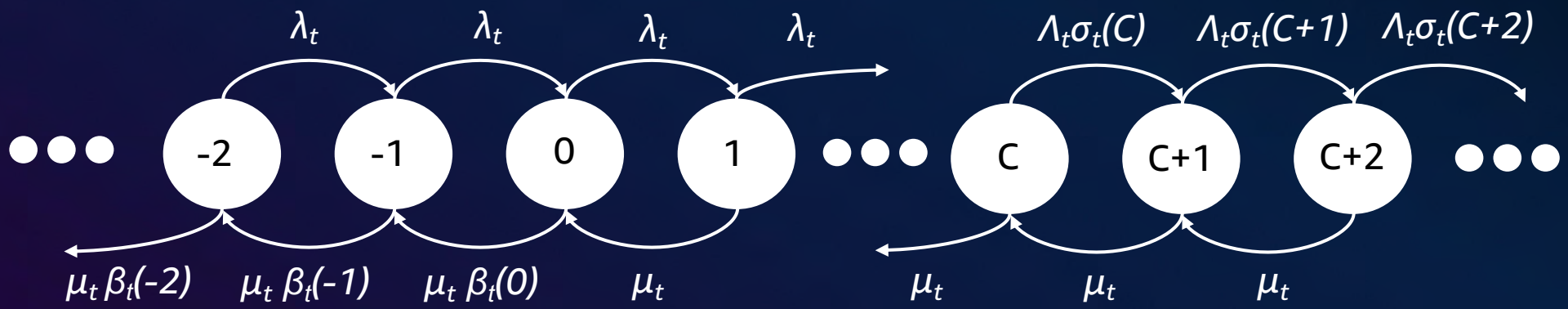
- 十分な予備容量を常に手元に持っておきます
- 障害時の積極的な監視をします
- ハードウェアに障害が発生しても必然的に冗長性が維持されます

保存されたデータの定期的な耐久性監査



99.999999999% の耐久性



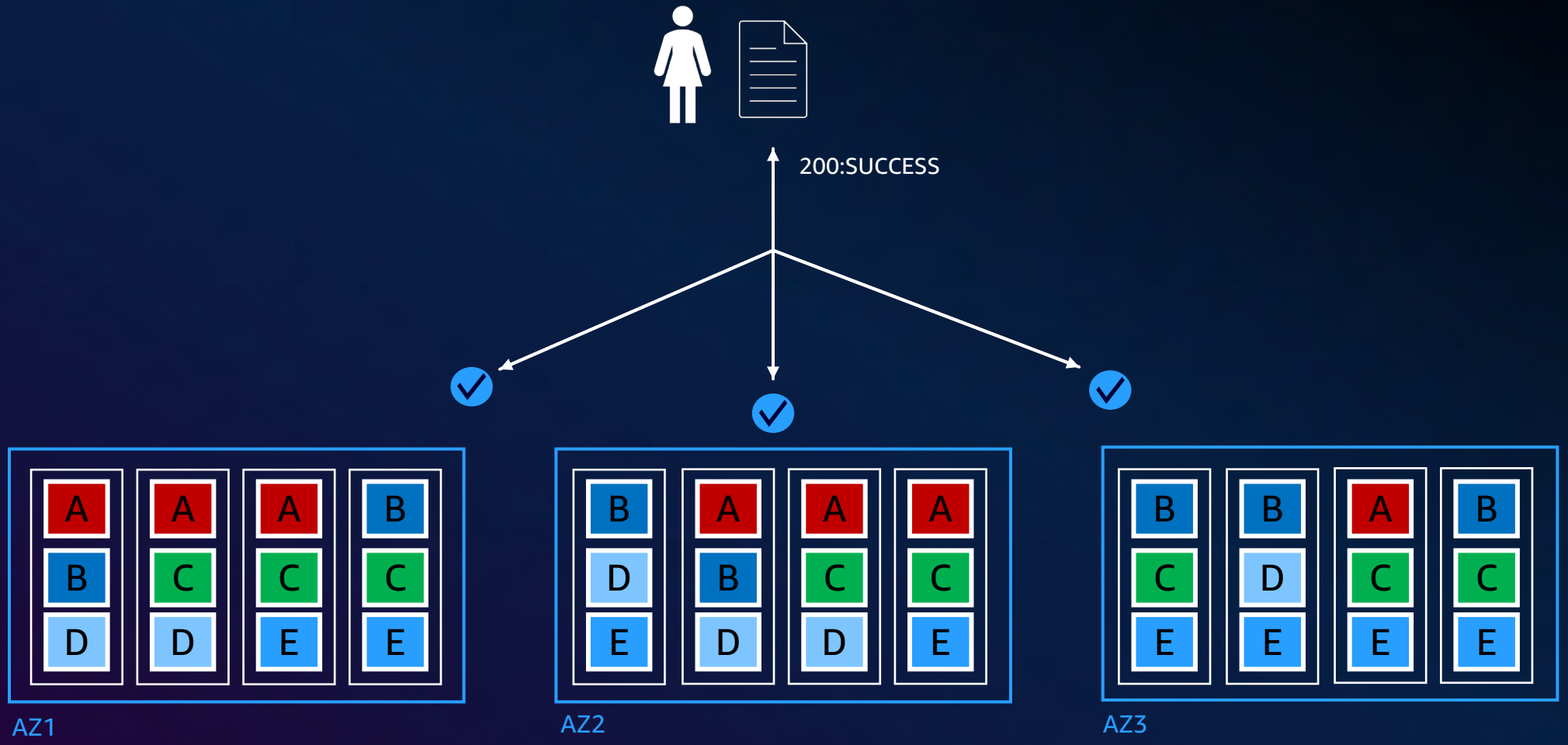


AZ 障害に対して

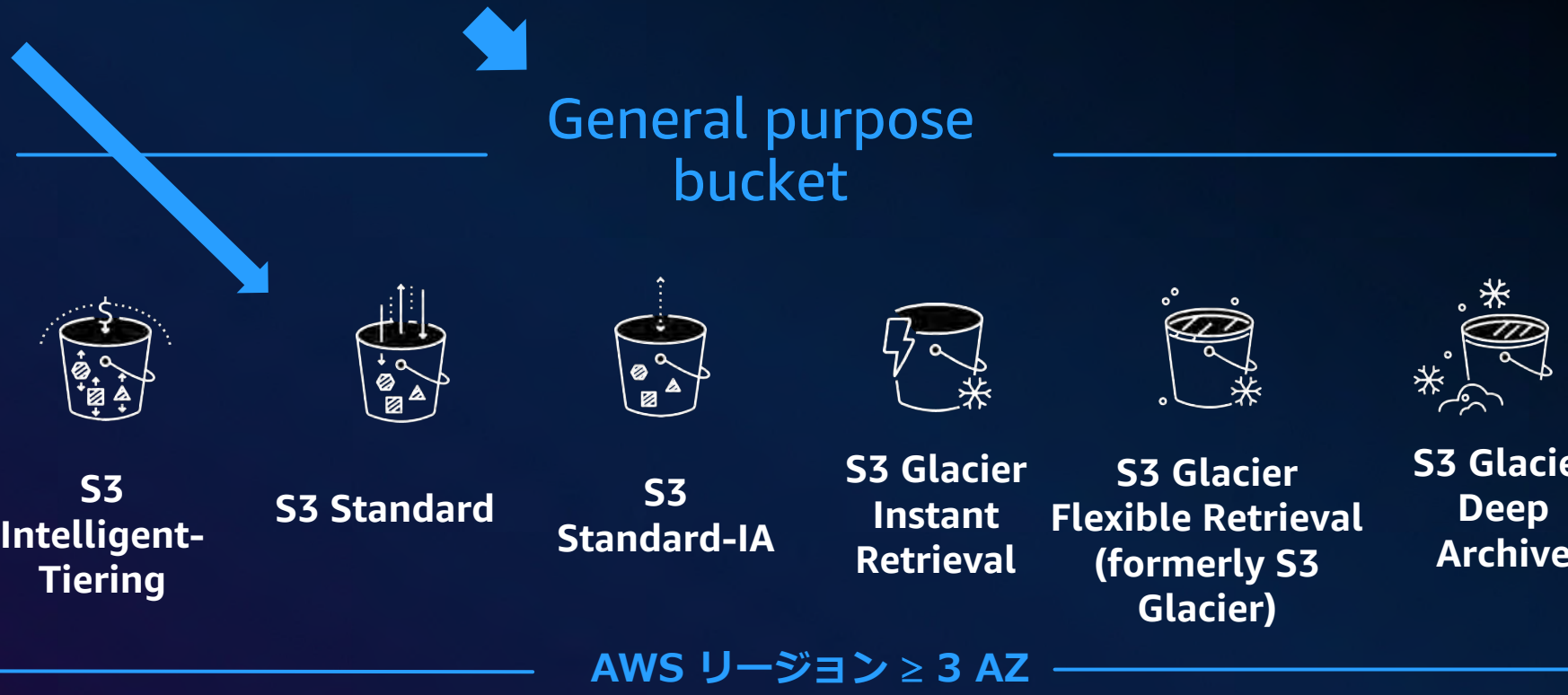
一つのゾーンの全体的または部分的な予期せぬ障害から、保管されているデータを保護する必要があります。いつでもどの施設でも障害が発生する可能性があり、その中でデータの耐久性を維持する必要があります

マルチ AZ がデフォルトです

マルチ AZ がデフォルトです



Amazon S3 ストレージクラス



Amazon S3 ストレージクラス

S3 EXPRESS ONE ZONE

フロントエンドの理解
インデックスの理解
ストレージ層の理解
Amazon S3 Express One Zone
誤削除への対策

Directory bucket

S3 Express One Zone

シングルAZ

General purpose bucket

S3 Intelligent-Tiering

S3 Standard

S3 Standard-IA

S3 Glacier Instant Retrieval

S3 Glacier Flexible Retrieval (formerly S3 Glacier)

S3 Glacier Deep Archive

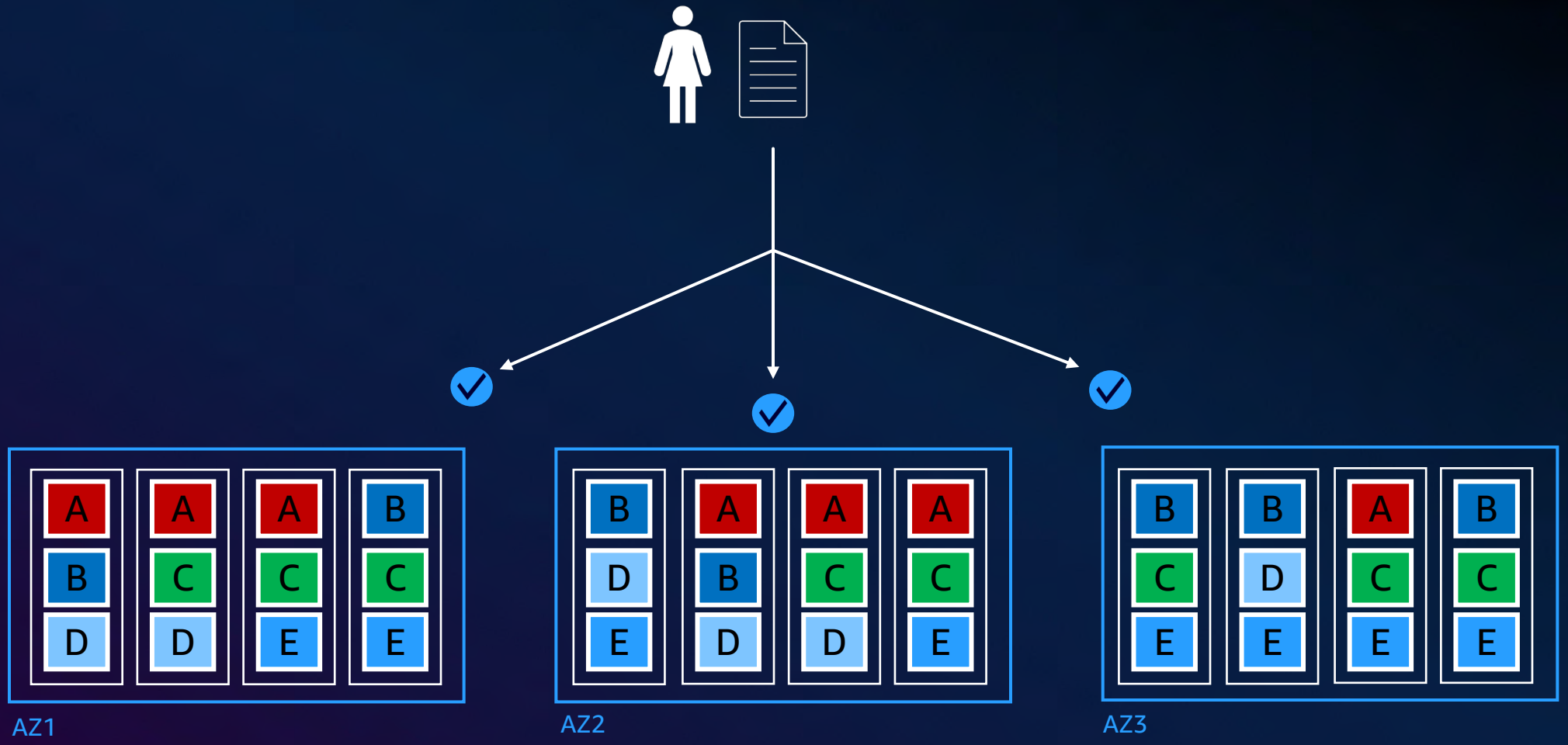
AWS リージョン ≥ 3 AZ

Amazon S3 Express One Zone

Amazon S3 Express One Zone では、アベイラビリティーゾーンの
全部または一部が失われると、データが失われる可能性があります

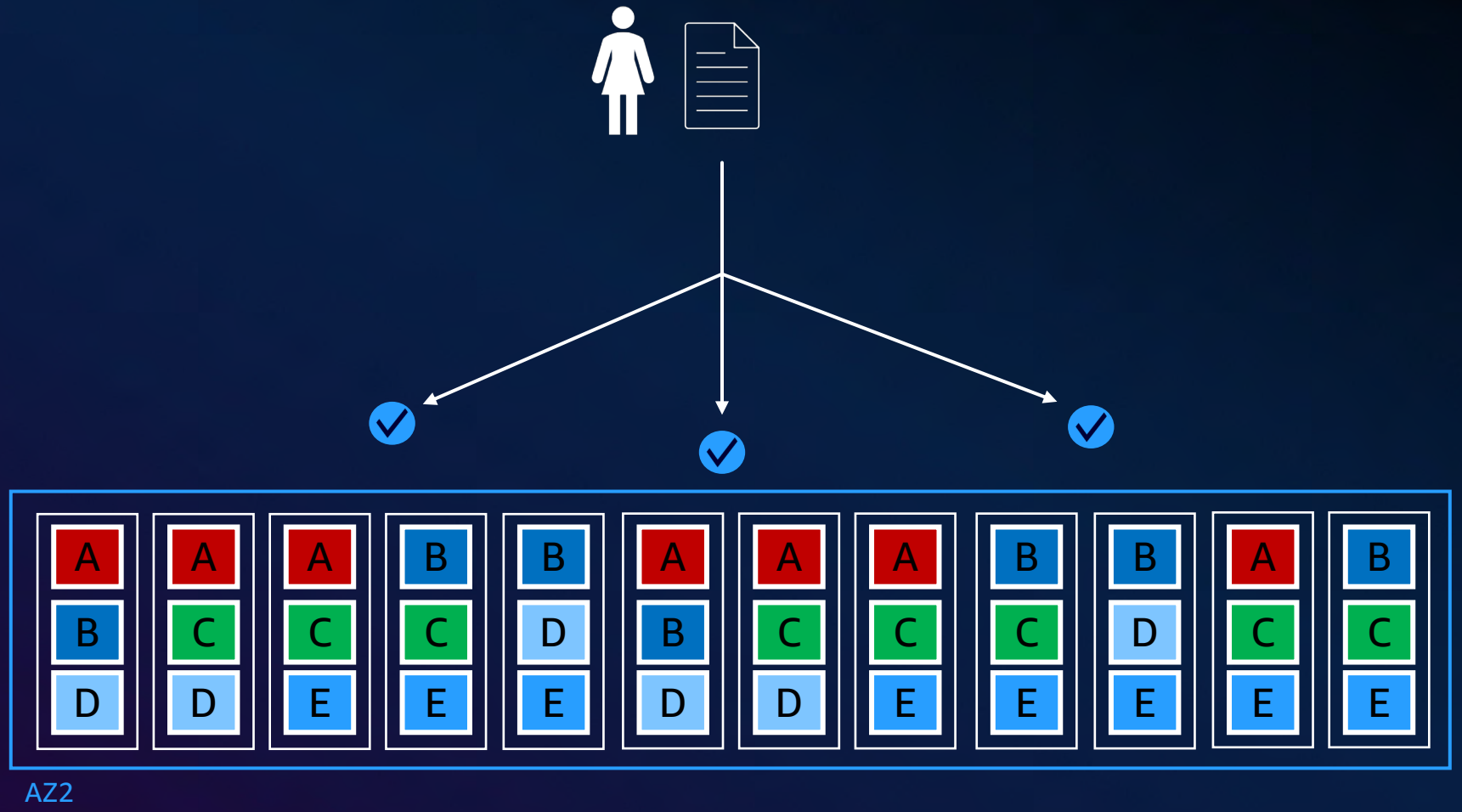
99.999999999% データ耐久性（おさらい）

S3 マルチ AZ でのデータ耐久性



99.999999999% データ耐久性

S3 EXPRESS ONE ZONE のシングル AZ でのデータ耐久性



99.999999999% データ耐久性

S3 EXPRESS ONE ZONEのシングルAZ でのデータ耐久性

エンドツーエンドでの
リクエストの整合性
チェック

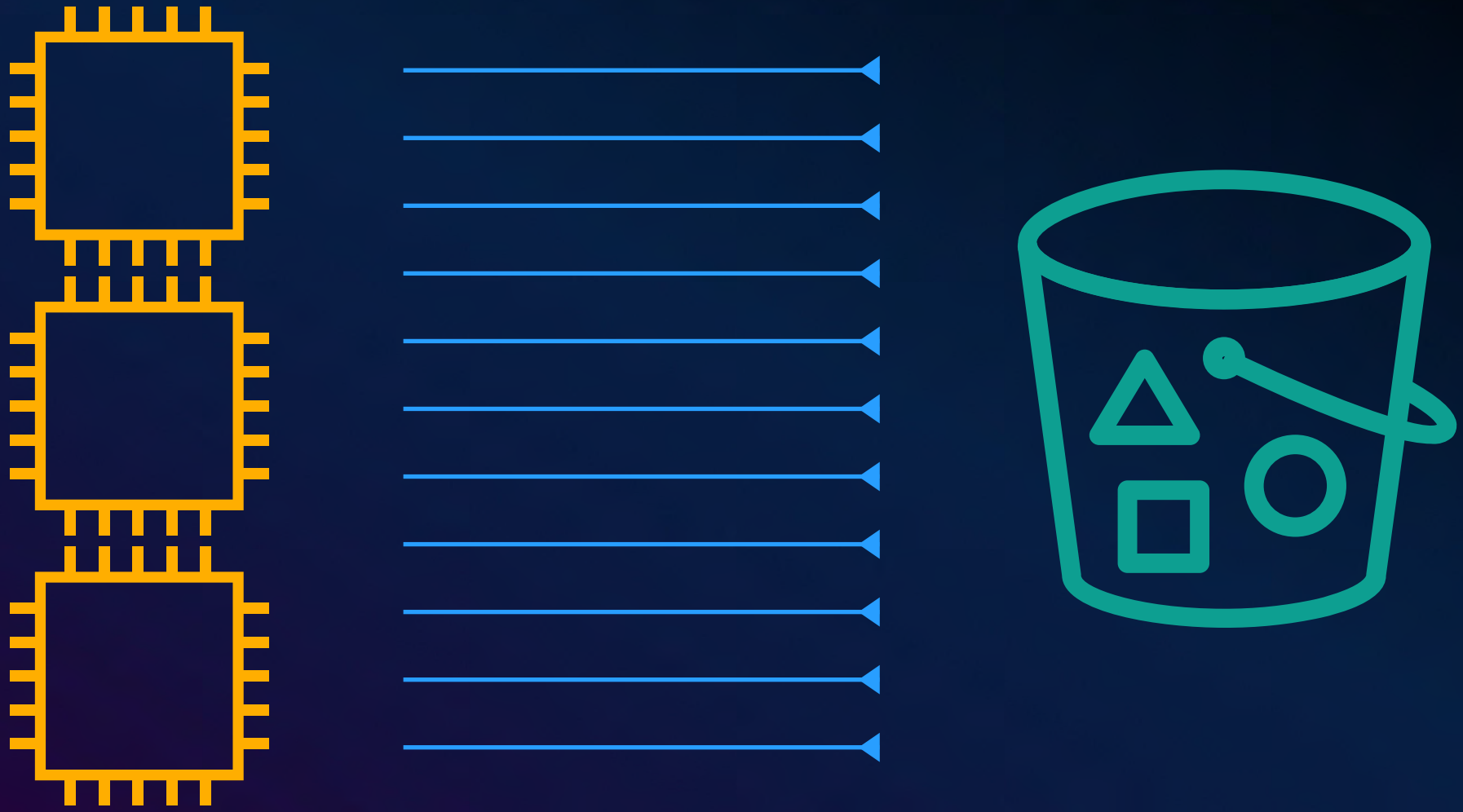
データは常に冗長性の
保たれたデバイスに
格納される

保存されたデータの
定期的な耐久性監査

AZの全体または部分障害で
データを失う可能性がある

なぜ S3 Express One Zone が開発されたのでしょうか？

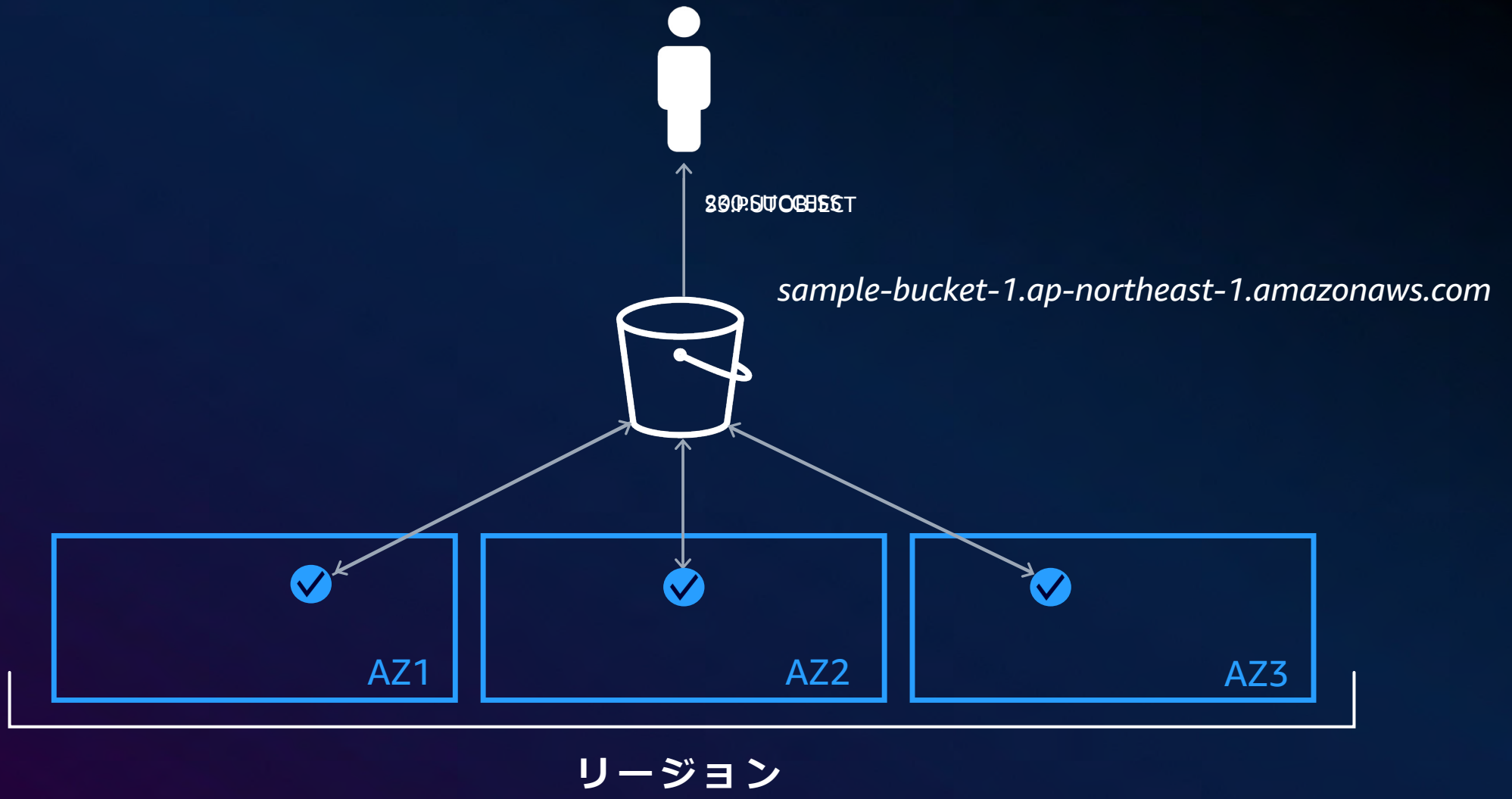
Amazon S3 自体は多くのリクエストを処理できる



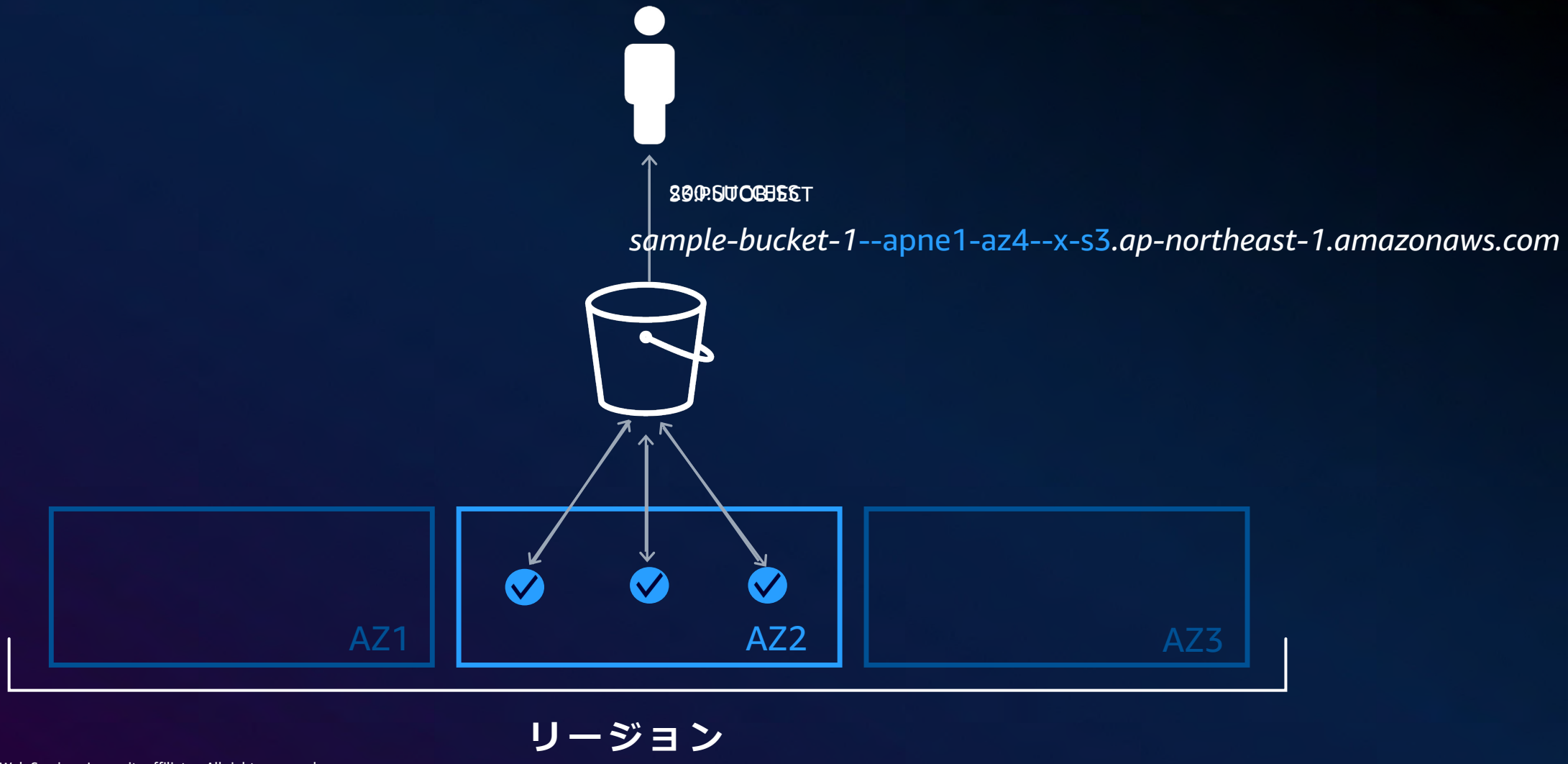
レイテンシを低減したいアプリケーションがある

では、どのようにして S3 Express One Zone で 高パフォーマンスを実現するのでしょうか

マルチ AZ のアーキテクチャ(おさらい)



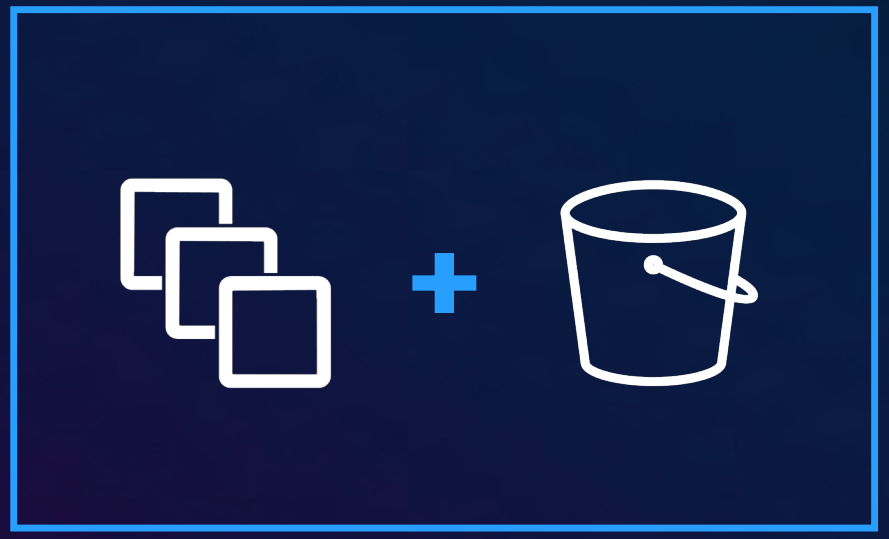
One Zone のアーキテクチャ



One Zone のアーキテクチャ

アーキテクチャーへの影響

アプリケーションで利用したい AZ



二つの考慮事項:

(1) AZ への配置によって、ネットワークレイテンシに影響があること（コストには影響しません）

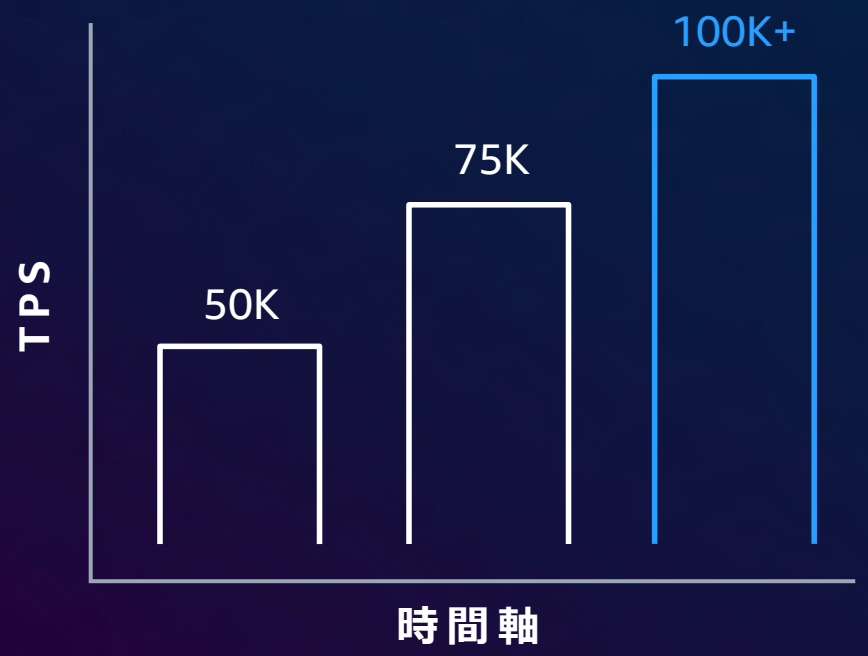
(2) One Zone のストレージには異なる耐久性モデルがあること

S3 directory bucket という新しいバケットタイプ

異なるスケーリングモデル

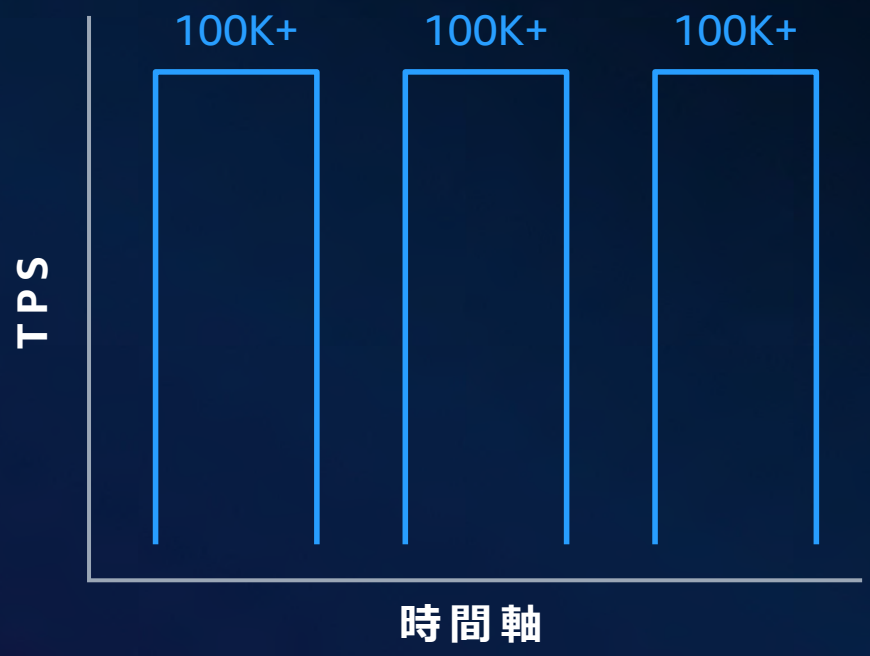
General purpose buckets

プレフィックス単位でスケーリングし
負荷がかかった状態で TPS が段階的に増える



Directory buckets

バケット単位でスケーリングし
始めから一定性能が提供される

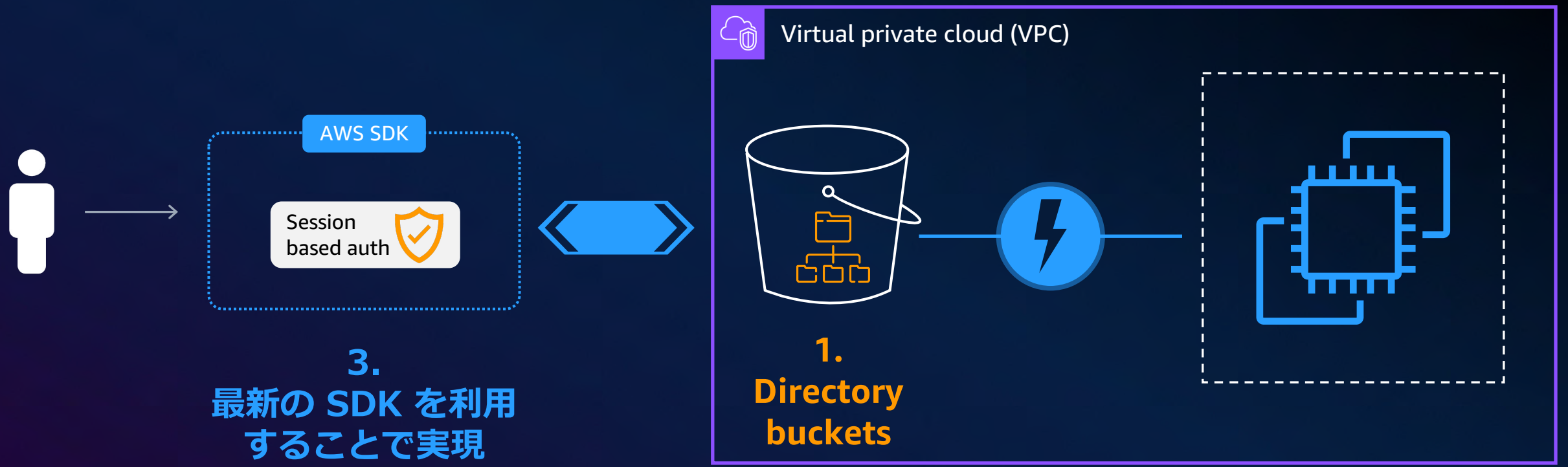


S3 Express One Zone のセキュリティモデル

バケットポリシーにセッションの作成を追加する

```
{

    "Version":"2012-10-17",
    "Statement":[
     {
       "Effect":"Allow",
       "Principal":{"AWS":"111122223333"},
       "Action":"s3express:CreateSession",
       "Resource":"*",
       "Condition": {
          "StringEquals": {"s3express:SessionMode":"ReadOnly"}
       }
     }
   ]
}
```

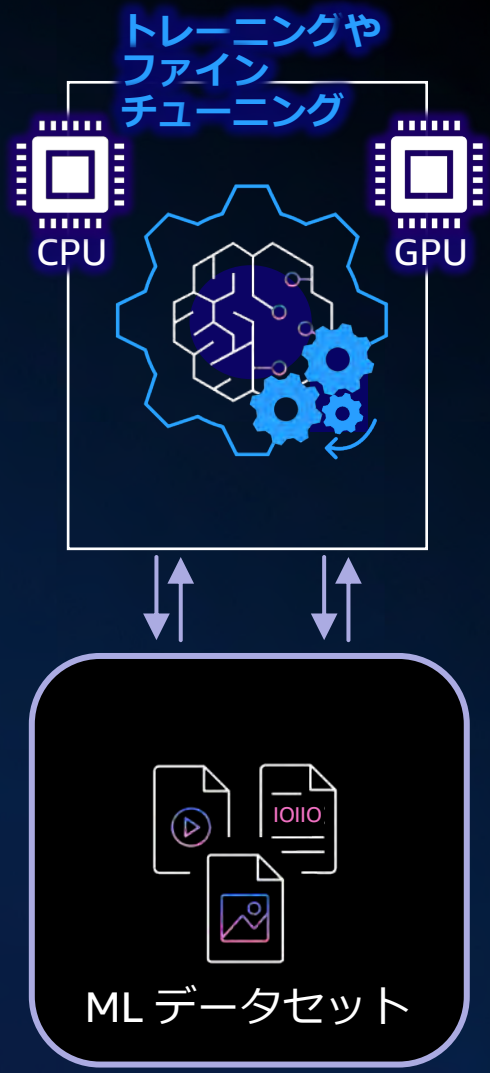
S3 Express One Zone 性能重視の要素



機械学習における「トレーニング」では、I/O 性能を重視

GPU や CPU を使い切るのは難しい一方で、活用されていなくてもコンピュートリソースには費用がかかります

I/O やスループットのなんらかのボトルネックが、不必要にトレーニングの時間を増やしてしまいます



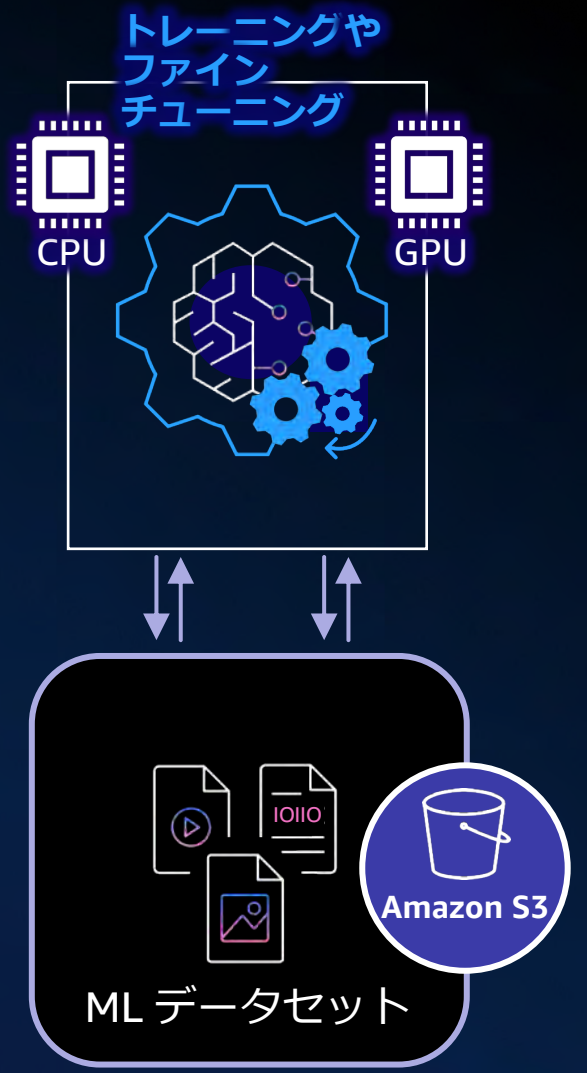
学習ジョブのストレージ性能に影響を与える要素

全体のデータセットのサイズや、全体のファイルの数

それぞれのファイルサイズ：小さいのか大きいのか

データの取り込み：あらかじめコピーしておくのか、
オンデマンドで逐次処理するか

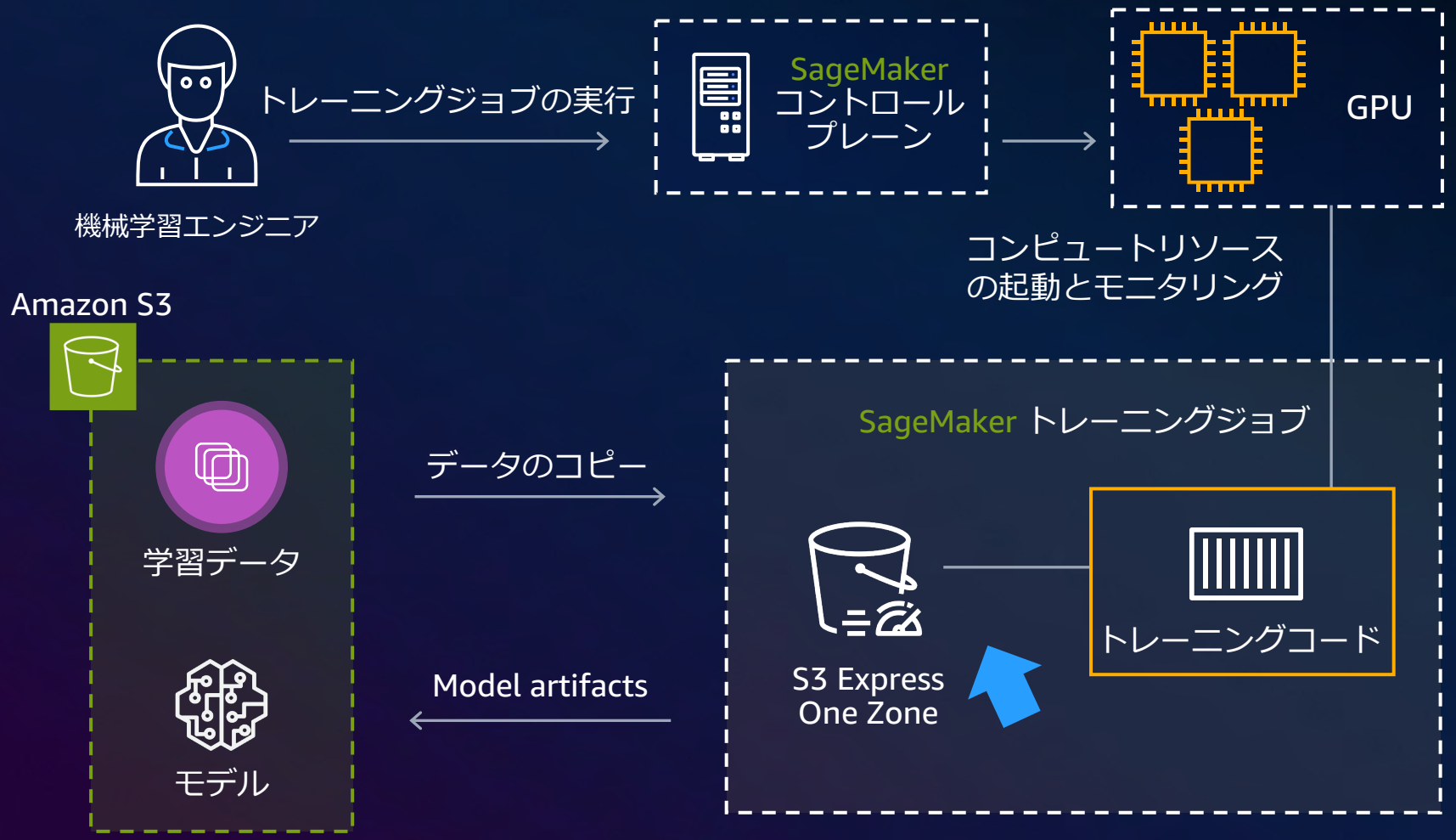
アクセスパターン：シーケンシャルまたはランダム



ワークロードが高速化されるシナリオ

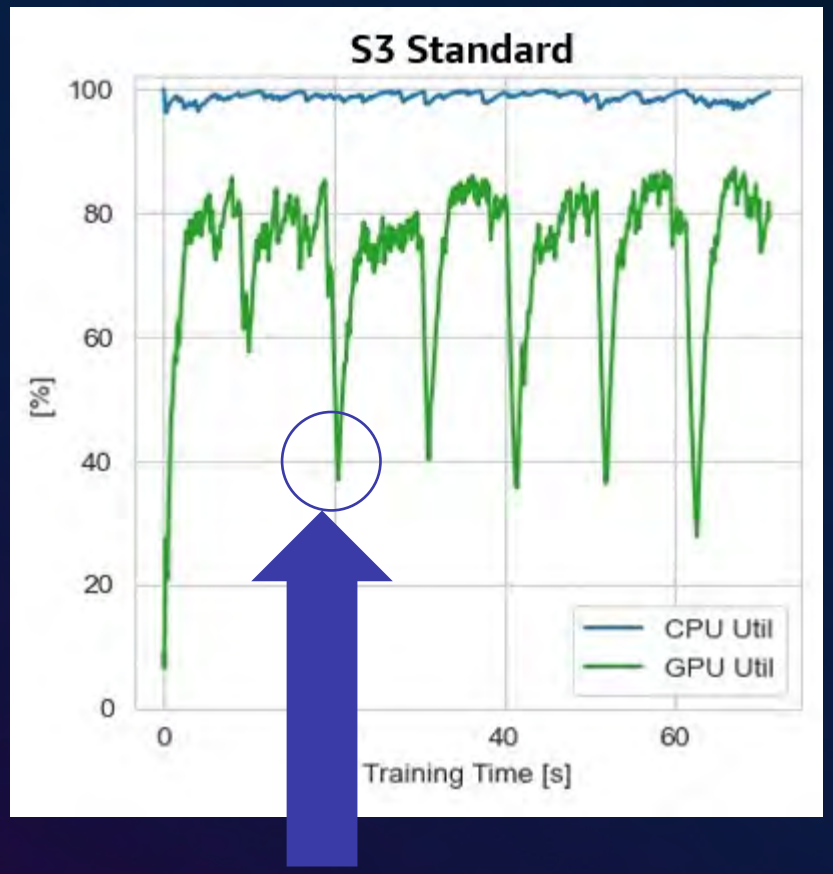
ジョブの合計実行時間が短縮されます

機械学習での S3 Express One Zone 活用例



- ✓ トレーニングジョブの高速化
- ✓ コンピューティングコストの低減

機械学習での S3 Express One Zone 活用例



GPU が、ストレージからデータを取得するのを待っている

ストレージのレイテンシが低く抑えられていることで、GPU の利用率が一貫して高い状態

クライアント

フロントエンド
Web サーバ, DNS, ネットワーク

インデックス
ストレージ層への Key/Value マップ

ストレージ層
媒体でのデータの冗長性確保

フロントエンドの理解

インデックスの理解

ストレージ層の理解

Amazon S3 Express One Zone

誤削除への対策

誤削除に対して

何かを削除するように言われたら、削除してしまいます

特に一括削除操作が原因で、Amazon S3 バケット内のデータを
誤って削除することは絶対に避けたいものです

誤削除に対して

緩和策として対応できる仕組み

S3 Versioning

S3 Replication

S3 Object Lock

Backups

上述の誤削除対策の機能は、General purpose bucket にて利用可能で、Directory bucket では利用できません

まとめ

フロントエンドの理解　-> リクエストの分散

インデックスの理解　-> キー命名のコツ

ストレージ層の理解　-> 99.999999999% 耐久性

Amazon S3 Express One Zone ストレージクラス

誤削除への対策 -> 永続性が大切なデータの配置

No compression algorithm for experience

経験を積むための圧縮アルゴリズムは
存在しない

Thank you!