AWS Black Belt Online Seminar

Amazon Kinesis Video Streams 応用編

三平 悠磨

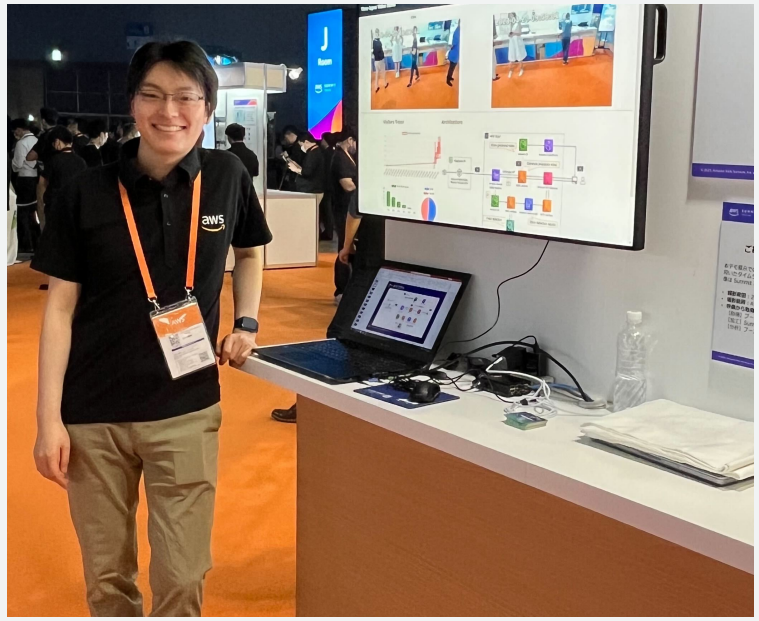
IoT Specialist Solutions Architect
2023/09

自己紹介

名前：三平 悠磨 (みひら ゆうま)

所属：技術統括本部 IoT ソリューション部

経歴：会話 AI 開発、家庭用ロボット開発



好きなAWSサービス：

Amazon Kinesis Video Streams, AWS IoT Greengrass

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は Twitter へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では 2023 年 09 月時点のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます
- 技術的な内容に関しましては、有料の AWS サポート窓口へお問い合わせください
- 料金面でのお問い合わせに関しましては、カスタマーサポート窓口へ
お問い合わせください (マネジメントコンソールへのログインが必要です)

本セミナーの対象者

本セミナーでは以下のような方を対象に、サービスを活用する上での Tips を紹介します。

- Amazon Kinesis Video Streams (KVS) を**既に利用**されており、**サービスをより上手く活用する方法**を検討されている方
- Amazon Kinesis Video Streams を**評価**しており、**パフォーマンスの改善方法やコスト構造への理解**を深めたい方
- Amazon Kinesis Video Streams を**これから利用予定**で、**よくある注意点**についてあらかじめ把握しておきたい方

➢ Amazon Kinesis Video Streams を初めて学ぶ方は、同月に公開される **Black Belt Amazon Kinesis Video Streams 基礎編** からご視聴ください。

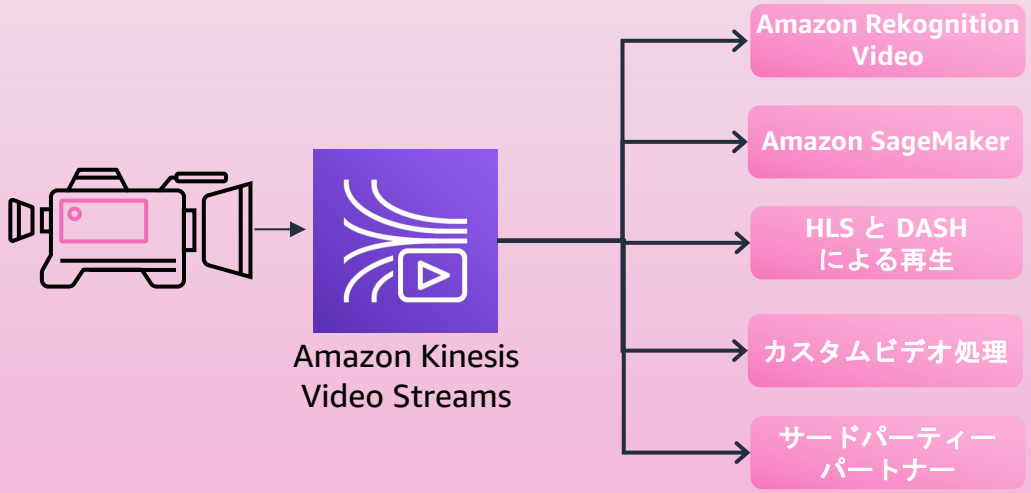
Kinesis Video Streams - 2 つのストリーミング方法

Streams

Kinesis Video Streams
何百万ものカメラデバイスからのセキュアなデータ取り込み

メディアを取り込み、保存、消費、
タイムインデックス付きメディアデータを再生

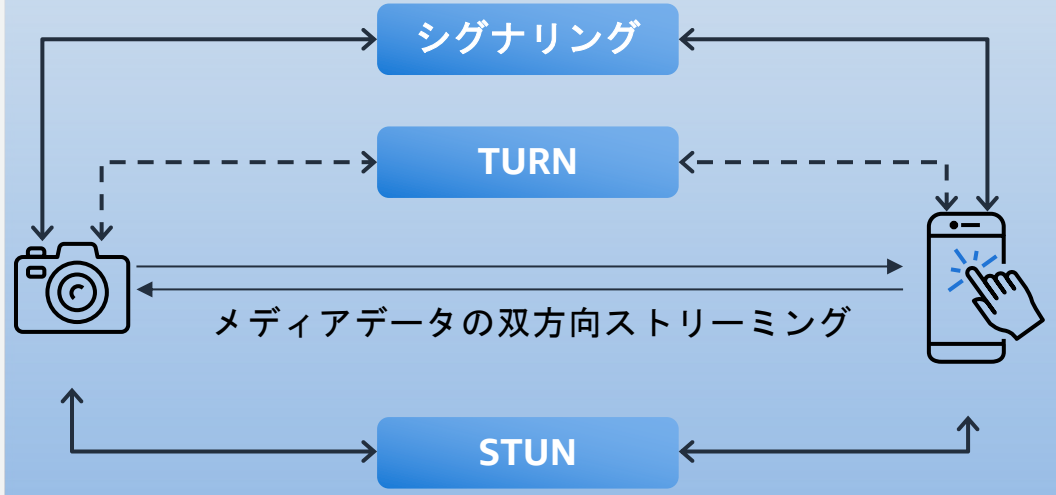
AI/ML サービスとの統合



WebRTC

Kinesis Video Streams
WebRTC による低遅延かつ双方向のメディアストリーミング

マネージドシグナリング、STUN、TURN サーバ

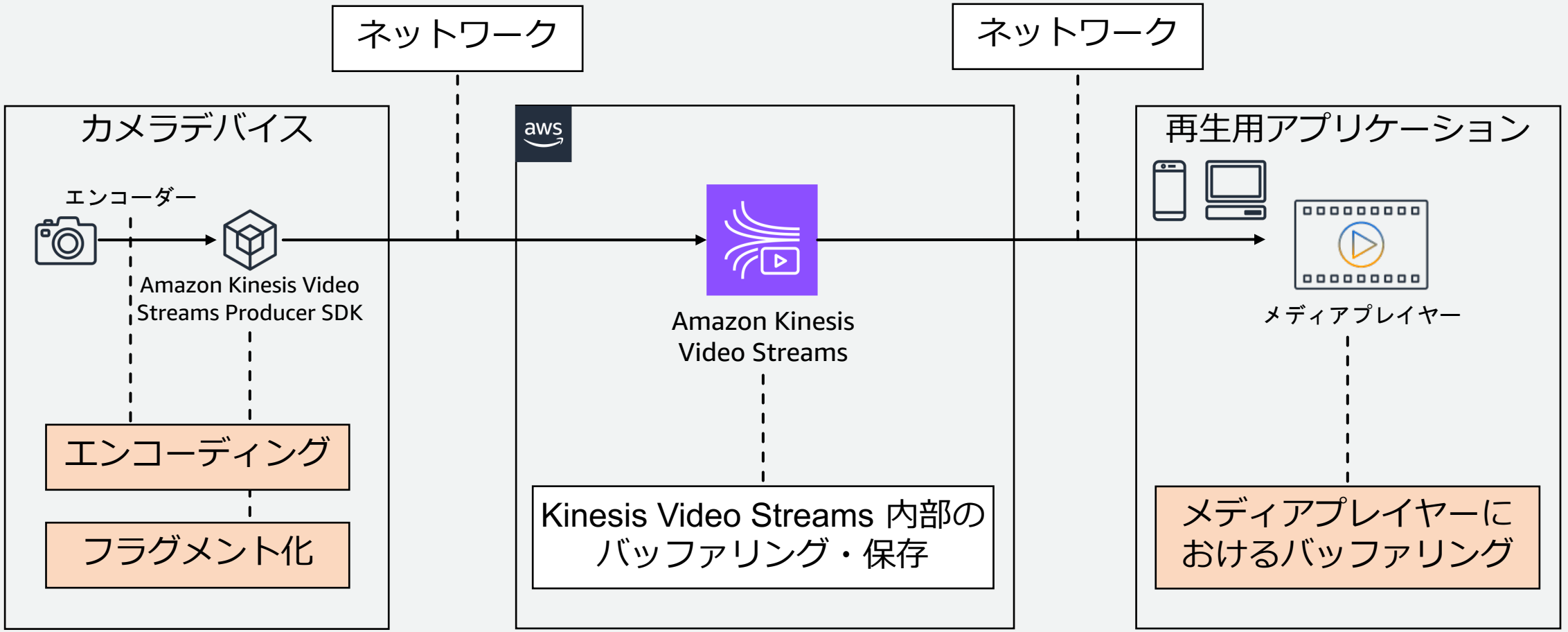


アジェンダ

- Tips #1 遅延の削減
- Tips #2 注意すべきサービスの仕様や上限
- Tips #3 コストの考え方
- Tips #4 WebRTC のトラブルシューティング
- Tips #5 GStreamer のよくある問題
- まとめ

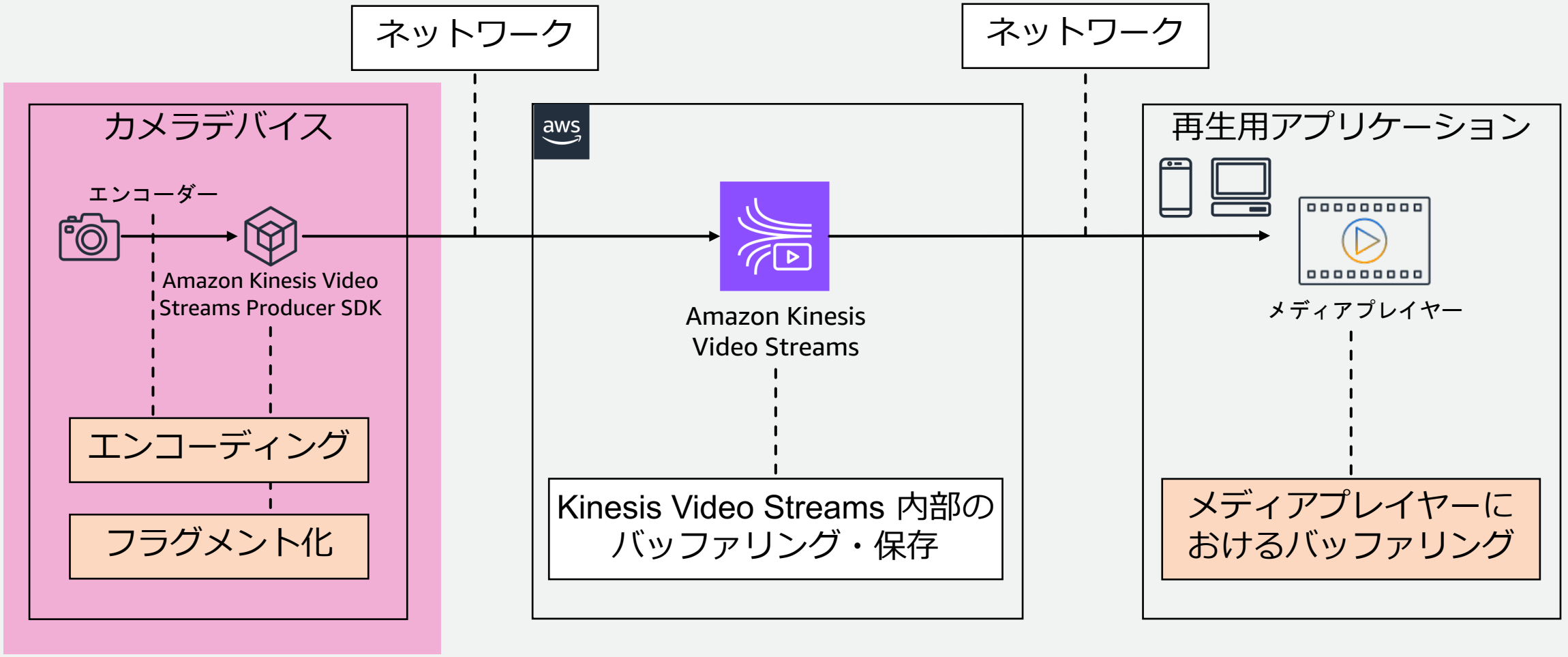
Tips #1 遅延の削減

Kinesis Video Streams における遅延の要因



https://aws.amazon.com/jp/blogs/news/how-to-reduce-video-latency-with-amazon-kinesis-video-streams-part-1/

Kinesis Video Streams における遅延の要因

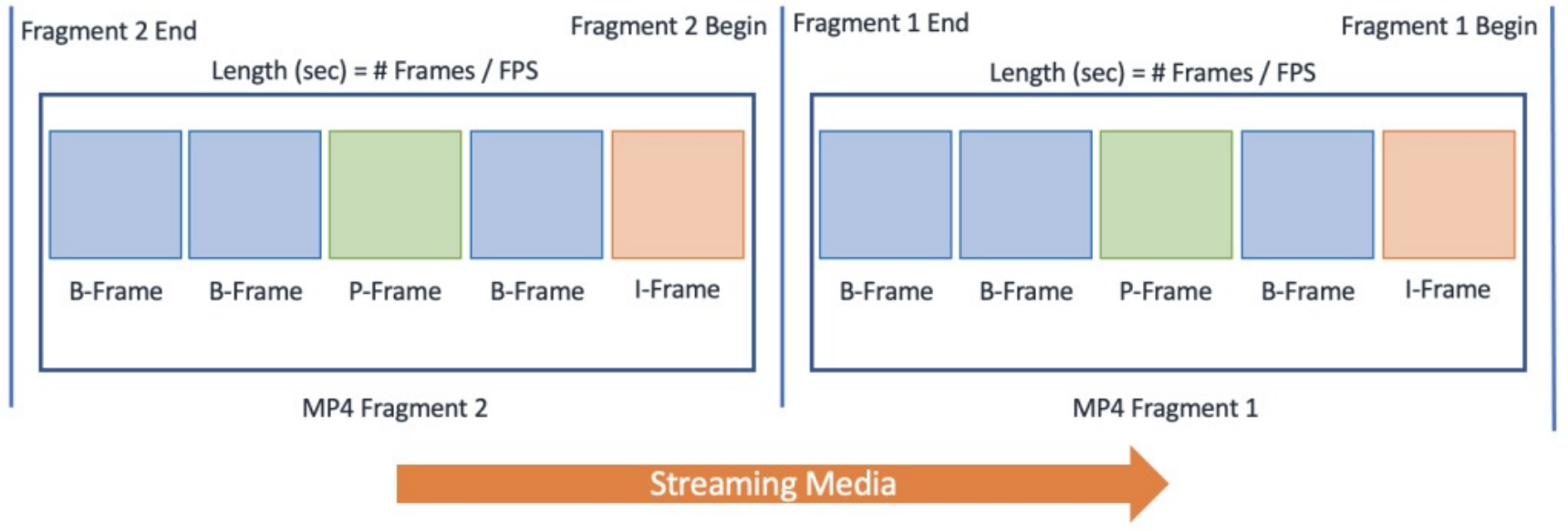


https://aws.amazon.com/jp/blogs/news/how-to-reduce-video-latency-with-amazon-kinesis-video-streams-part-1/

エンコードとフラグメント長 (1/2)

Streams

- Kinesis Video Streams ではフラグメント単位で動画をストリーミング・管理するため、**フラグメント長**が全体の遅延に影響する
- フラグメント長は**キーフレーム間隔**および **SDK の設定**に左右される
- 各フラグメントは必ず**キーフレーム(I-フレーム)から開始**する



エンコードとフラグメント長 (2/2)

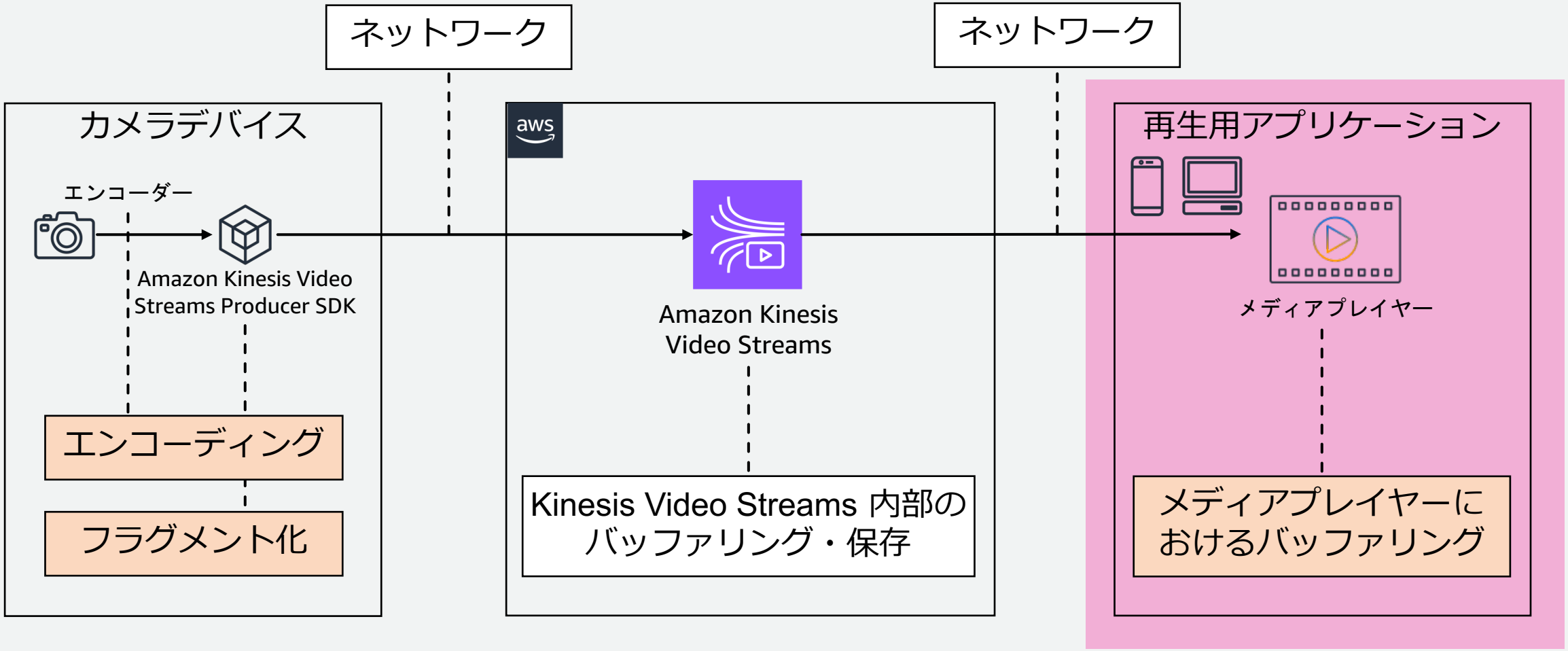
Streams

- 期待するフラグメント長になるよう、**エンコーダーのキーフレーム間隔を設定**する必要がある
  - 例：30fps のビデオで、フラグメント長を 1.5 秒にしたい場合
  - (最大の)キーフレーム間隔を 30 x 1.5 = 45 に設定する
- 設定方法
  - 例: GStreamer の x264enc (ソフトウェアエンコーダー)の場合、key-int-max の値で設定できる
- **KVS Producer SDK** 側でも**フラグメント長 (fragment-duration) を設定**する
- 注意点
  - ビットレート(データ量)とのトレードオフになるため、ユースケースに応じて適切な値を選択する
  - ハードウェアエンコーダーやカメラ自体がエンコーディングを行なっている場合、設定項目が存在しない場合もある

https://gstreamer.freedesktop.org/documentation/x264/index.html?gi-language=c#x264enc:key-int-max

aws

Kinesis Video Streams における遅延の要因

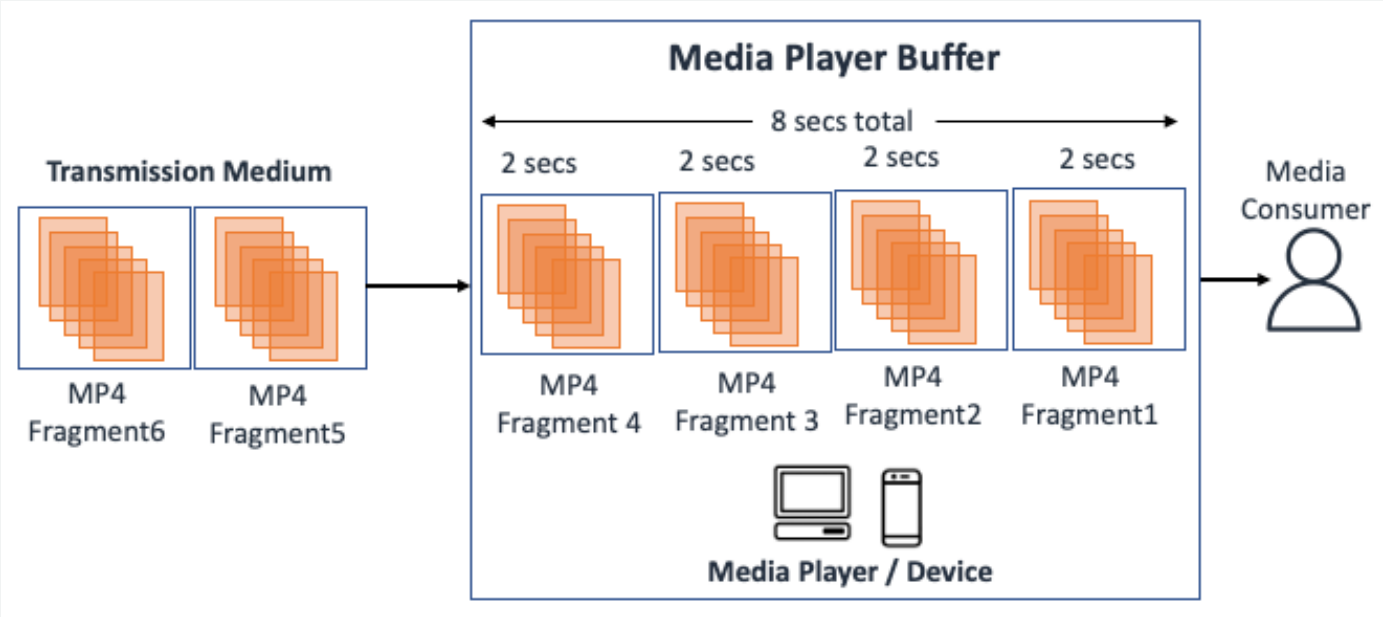


https://aws.amazon.com/jp/blogs/news/how-to-reduce-video-latency-with-amazon-kinesis-video-streams-part-1/

メディアプレイヤーのバッファリング (1/2)

Streams

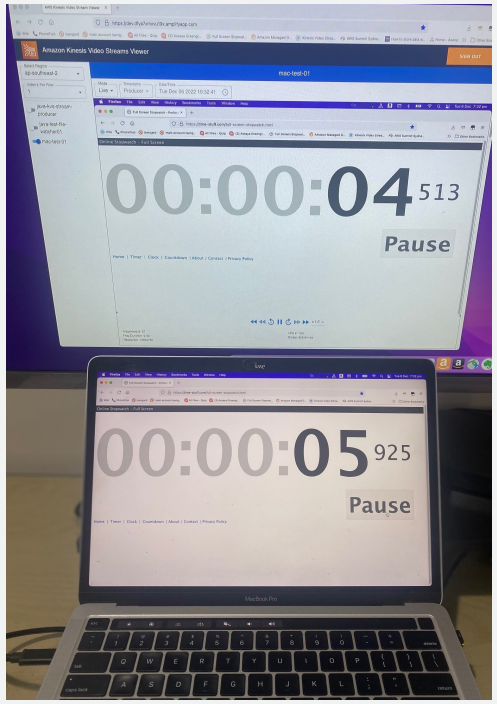
- メディアプレイヤーは通常、ネットワークが不安定な際にビデオの品質が低下するのを避けるため、ある程度の数のフラグメントを**バッファリング**する
- **ライブ再生用に最適化されていない設定**では、**全体の遅延の大半**がメディアプレイヤーのバッファリングによるものになる場合もある



メディアプレイヤーのバッファリング (2/2)

Streams

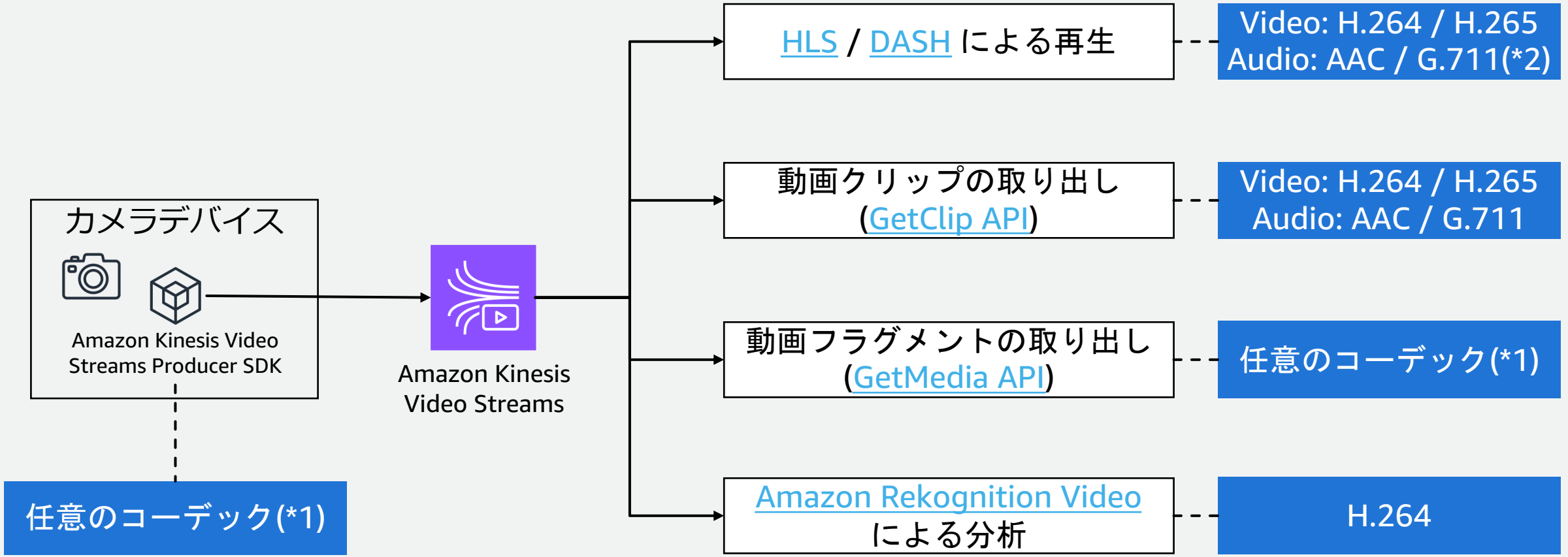
- **KVS Web Viewer** のデモアプリケーションの場合
  - React-Player を利用し、HLS.js をデフォルトでロード
  - HLS.js のライブ再生用の設定をデフォルト値から変更
    - liveSyncDurationCount: 3 → 2
    - liveMaxLatencyDurationCount: Inf → 3
  - 常に最新の映像にキャッチアップするよう、再生速度を x1 よりもわずかに上に設定する
    - ただし、映像が途切れるなどのトレードオフが発生する恐れがある
- 下記のブログ記事では、End-to-end の遅延を**約 1.4 秒**まで削減できた設定例を紹介



https://aws.amazon.com/jp/blogs/news/how-to-reduce-video-latency-with-amazon-kinesis-video-streams-part-2/

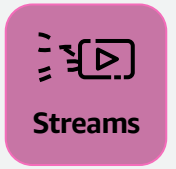
Tips #2 注意すべき サービスの仕様や上限

サポートするコーデック



*1: Matroska のコンテナ形式がサポートするもの
*2: DASH の場合のみ A_MS/ACM をサポート

注意すべきサービスの上限 (Streams)



上限緩和申請が可能なソフトリミットを [s]、上限緩和ができないハードリミットを [h] と表記しています

コントロールプレーン API の制限

ストリーム数: アカウント・リージョンあたり 5,000 もしくは 10,000ストリーム [s]

デバイス台数に応じて上限緩和申請

メディアとアーカイブメディア API の制限

PutMedia - 5 TPS [h] (ストリームレベル)、帯域幅制限 100 Mbps [s]、フラグメント継続時間 1～20秒 [h]

GetMedia - 5 TPS [h] (ストリームレベル)。同時にメディアストリームからコンテンツを受信できるクライアントは 3 つまで [s]

GetHLSStreamingSessionURL - 25 TPS [h] (ストリームレベル)

再生時のクォータ: フラグメントベースの制限となっており、**通常100程度のクライアントから同時視聴が可能**

https://docs.aws.amazon.com/kinesisvideostreams/latest/dg/limits.html

注意すべきサービスの上限 (WebRTC)

上限緩和申請が可能なソフトリミットを [s]、上限緩和ができないハードリミットを [h] と表記しています

コントロールプレーン API の制限

シグナリングチャネル数: アカウント・リージョンあたり 5,000 もしくは 10,000 チャネル [s]

デバイス台数に応じて上限緩和申請

シグナリング API サービスの制限

ConnectAsMaster – シグナリングチャネルあたりの Master 接続の際台数: 1 [h]

ConnectAsViewer - **シグナリングチャネルあたりの Viewer 接続の際台数: 10 [s]**

TURN サービスの制限

ビットレート - 5Mbps [h]

https://docs.aws.amazon.com/ja_jp/kinesisvideostreams-webrtc-dg/latest/devguide/kvswebrtc-limits.html

動画ストリーミングサービスの使い分け

Amazon Kinesis Video Streams

- コネクテッドカメラデバイスからクラウドへセキュアにストリーミング
- デバイス台数の増加に対するスケーリング
- 再生、機械学習による分析、その他の処理をリアルタイムもしくはオンデマンドで実施
- WebRTC による低遅延の双方向ストリーミング
- 例: スマートホームカメラ、監視カメラ、車載カメラ、ロボット、ドローン、産業用の自動処理

AWS Elemental Media Services & Amazon CloudFront

- 放送グレードの品質および柔軟な配信要件に対応可能
- ビルディングブロック、配信要件に応じて詳細な設定変更が可能
- HEVC、4K対応
- 例: 放送グレードのメディア配信、パッケージング、広告挿入、 DRM、スケジューラー、 DVR/タイムシフト

Amazon Interactive Video Sercvice

- 素早く簡単にセットアップできるマネージドライブストリーミングサービス
- OTT の知識や経験がなくとも、数分でワールドワイドにライブ配信基盤を展開可能
- 超低遅延配信、Timed Metadata API、Player SDK を利用することで双方向コミュニケーションを容易に実現可能
- Stage によるマルチホスト配信、1秒未満の遅延の配信が可能
- 例: インタラクティブなライブ配信

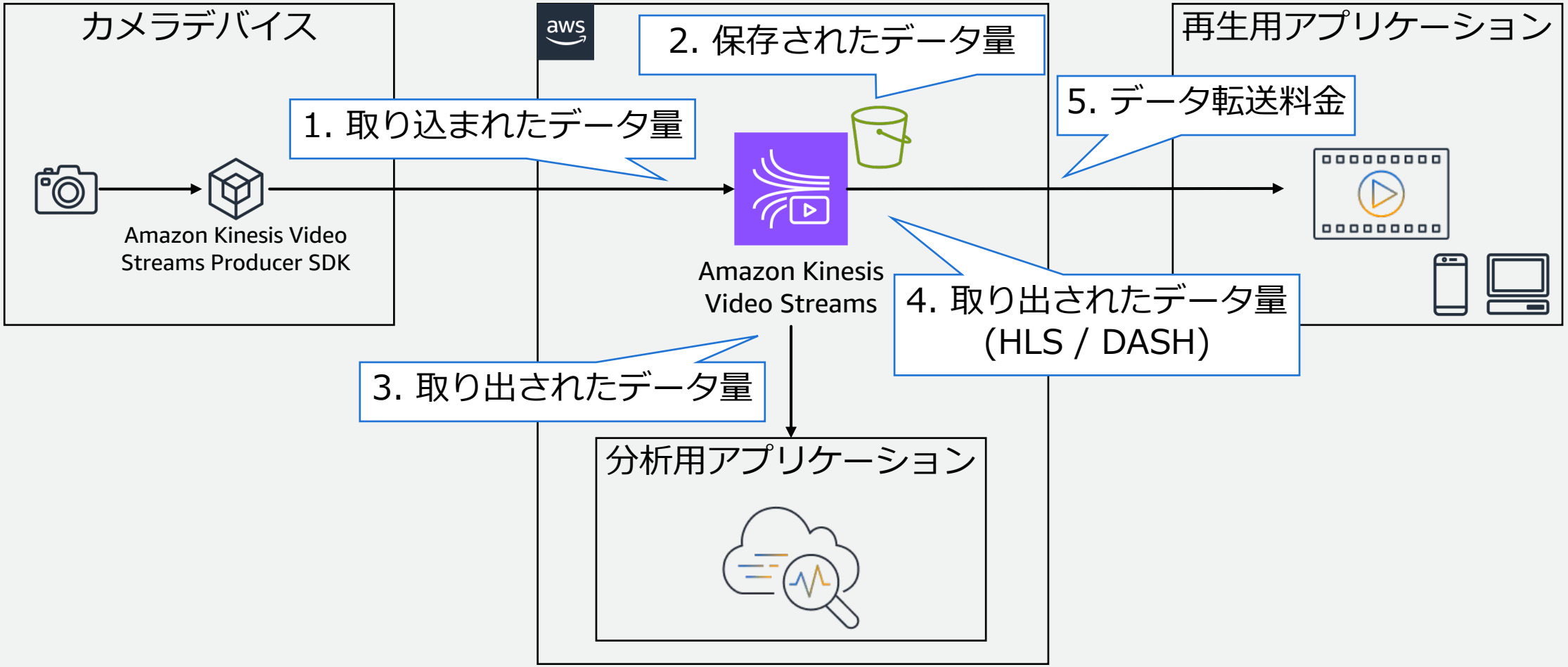
Amazon Chime / Amazon Chime SDK

- 複数の話者が登場する M：N のビデオ通話用途
- 双方向の円滑な会話のための1秒未満配信遅延
- 例: ビデオ会議

https://aws.amazon.com/jp/blogs/news/choose-the-right-aws-video-service-for-your-use-case/

Tips #3 コストの考え方

Kinesis Video Streams の料金構成



https://aws.amazon.com/jp/kinesis/video-streams/pricing/

ユースケース毎のコストファクター

Streams

ユースケース	ファクター				料金体系
取り込み	ビットレート	× カメラあたり取り込み時間	× 台数		1. 取り込まれたデータ量
保存	ビットレート	× カメラあたり取り込み時間	× 台数	× 保存期間	2. 保存されたデータ量
ダウンロード・分析	ビットレート	× カメラあたりダウンロード・分析時間	× 台数		3. 取り出されたデータ量 5. データ転送 (AWS 外へのダウンロードの場合)
再生 (ライブ/オンデマンド)	ビットレート	× カメラあたり再生時間	× 台数		4. 取り出されたデータ量 (HLS/DASH) 5. データ転送

https://calculator.aws/#/addService/KinesisVideoStreams

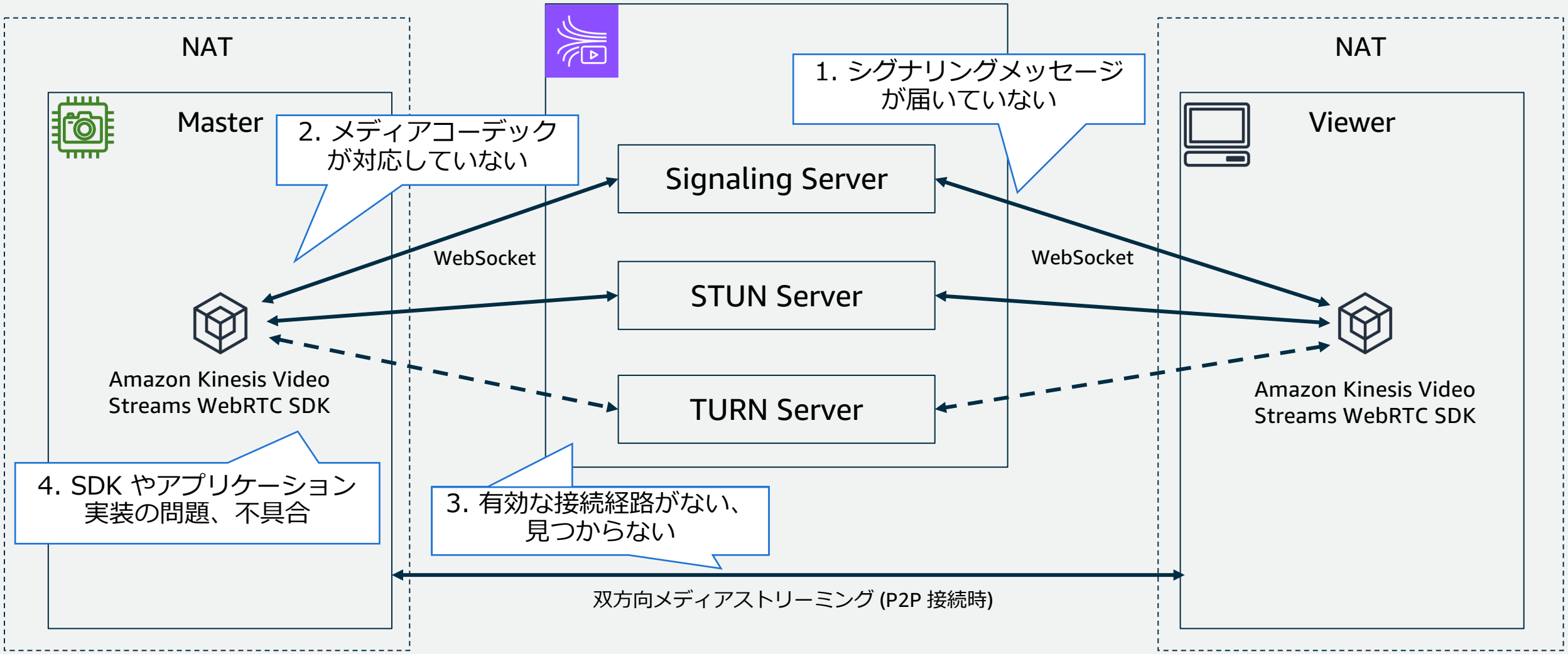
ユースケースに応じてコスト最適化する方法

Streams

- **全体コストの最適化 (ビットレートの削減)**
  - ユースケースにおいて必要最低限の解像度、フレームレートを利用
  - 高圧縮なコーデック/設定の利用 (各 API やプレイヤーがサポートするコーデックに注意)
- **長期保存が必要な場合**
  - 長期保存(半年、数年など)が必要だが、即時の取り出しや再生が不要なデータは、KVS から動画ファイル形式で取り出し、安価なストレージ (Amazon S3 Glacier など)に格納することを検討
- **再生時間が多い場合**
  - ライブ再生の場合 Kinesis Video Streams WebRTC を併用して P2P で映像視聴
  - 非アクティブ時に静止画表示などのアプリケーション側での対応
  - 連続再生時間・総再生時間に上限を設ける、再生時間に応じた課金プラン

Tips #4 WebRTC のトラブルシューティング

WebRTC の接続における問題のよくある原因



WebRTC のトラブルシューティング

前ページの**原因とそれに対応する確認・切り分け方法**

ログ・メトリクスの確認

確認箇所	対応する原因
WebRTC SDK で DEBUG レベルのログを出力する	1, 2, 3, 4
パケットキャプチャを確認する	1, 3
Chrome や Firefox ブラウザの場合は WebRTC Internals を確認する	1, 2, 3
CloudWatch メトリクスを確認する	1

下記ドキュメントを参考に**問題の切り分け**を行う
https://docs.aws.amazon.com/ja_jp/kinesisvideostreams-webrtc-dg/latest/devguide/troubleshooting.html

問題の切り分け

切り分けポイント	対応する原因
KVS WebRTC のエンドポイントへ接続できているか	1, 3
TURN が有効化されているか	3
SDP Offer や Answer が送信されているか、届いているか	1
ICE Candidate (接続経路の候補)が生成されているか	3
どの ICE Candidate Pair が選択されているか	3
別のネットワークを試す (特に VPN やコーポレートネットワークに接続している場合)	1, 3
別のブラウザやデバイスで試す	2, 4
サンプルアプリケーションを試す	4

WebRTC SDK (JavaScript) のログの例


```
[2023-07-14T05:07:35.947Z] [INFO] [VIEWER] Sending SDP offer
[2023-07-14T05:07:35.949Z] [DEBUG] SDP offer: { "type": "offer", "sdp":
"v=0¥r¥no=- 4825670103251983856 2 IN IP4 127.0.0.1¥r¥n" ...}

...

[2023-07-14T05:08:24.836Z] [INFO] [VIEWER] Received SDP answer
[2023-07-14T05:08:24.836Z] [DEBUG] SDP answer: { "type": "answer", "sdp":
"v=0¥r¥no=- 556581638 2 IN IP4 127.0.0.1¥r¥ns=-¥r¥nt=0 0¥r¥na=group:BUNDLE 0 1
2¥r¥n" ...}

...

[2023-07-14T05:07:36.149Z] [INFO] [VIEWER] Generated ICE candidate
[2023-07-14T05:07:36.149Z] [DEBUG] ICE candidate: { "candidate":
"candidate:1778954504 1 udp 8266495 123.45.67.89 57147 typ relay raddr 0.0.0.0
rport 0 generation 0 ufrag bBPz network-id 1 network-cost 10", … }

...

[2023-07-14T05:08:25.143Z] [DEBUG] Chosen candidate pair (video): { "local":
{ "candidate": "candidate:2452868866 1 udp 1685987071 123.45.67.89 50543 typ
srflx raddr 192.168.0.112 rport 64060 generation 0 ufrag 2JiF network-id 1
network-cost 10", "sdpMid": "", "sdpMLineIndex": 0, "usernameFragment": "2JiF" },
"remote": { "candidate": "candidate:2 1 udp 1694498815 12.34.56.78 37601 typ
srflx raddr 0.0.0.0 rport 0 generation 0 ufrag yUNE network-cost 999", "sdpMid":
"", "sdpMLineIndex": 0, "usernameFragment": "yUNE" } }
```

送信した SDP Offer

受信した SDP Answer

ICE Candidate
(接続経路の候補)

選択された接続経路

https://awslabs.github.io/amazon-kinesis-video-streams-webrtc-sdk-js/examples/

Tips #5 GStreamer のよくある問題

GStreamer とは

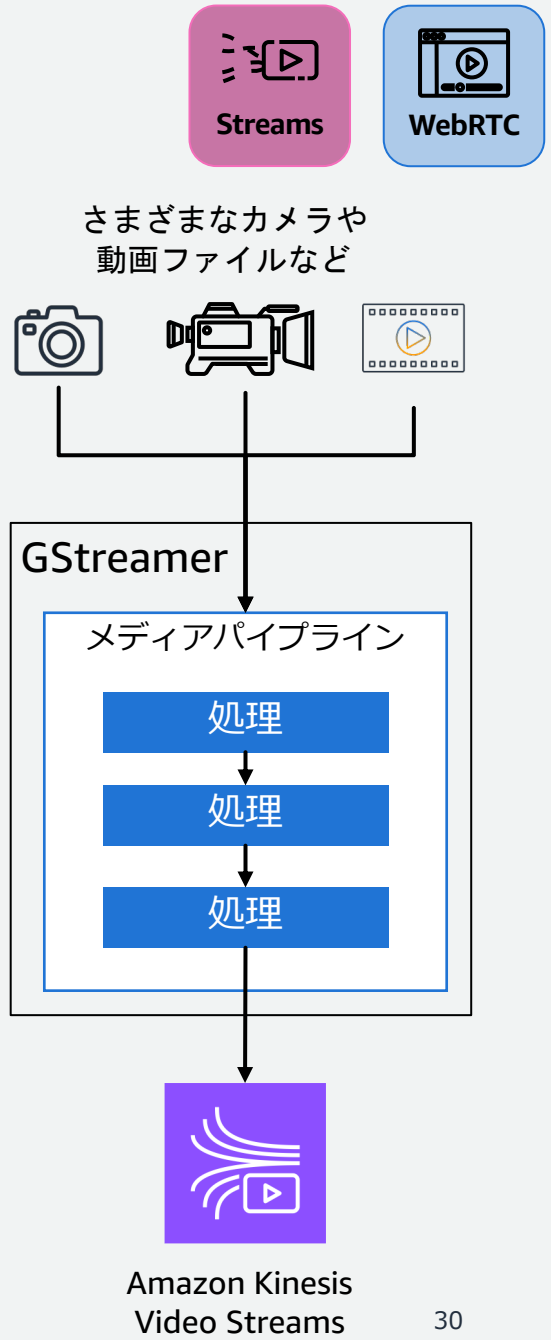
GStreamer は、**ストリーミングメディアパイプライン**を作成するためのオープンソースフレームワーク。

GStreamer を使用すると、**メディアをある形式で読み取り、何らかの方法で処理またはフィルタリングし、別の形式や/またはプロトコルでエクスポート**するアプリケーションを構築できる。

GStreamer は、**コミュニティやベンダーがサポートするオープンなプラグイン群**に基づいており、ユーザー設定可能なメディアパイプラインのエレメントをそれぞれが提供している。

Kinesis Video Streams では GStreamer を利用してさまざまなソースから映像をストリーミングできるよう、**プラグインやサンプルアプリケーション**を提供している。

https://gstreamer.freedesktop.org/
https://docs.aws.amazon.com/ja_jp/kinesisvideostreams/latest/dg/examples-gstreamer-plugin.html
https://github.com/awslabs/amazon-kinesis-video-streams-webrtc-sdk-c#running-the-samples

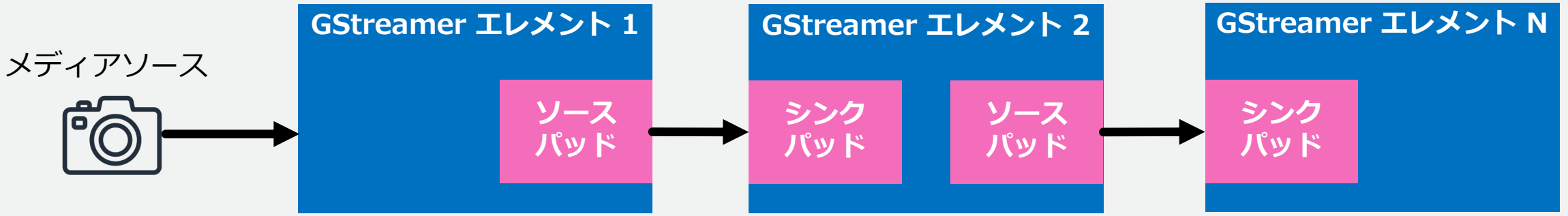


GStreamer エレメントとパッド

エレメント: メディアパイプラインを構築するための**ビルディングブロック**

パッド: GStreamer エレメント同士が通信する**インターフェース**

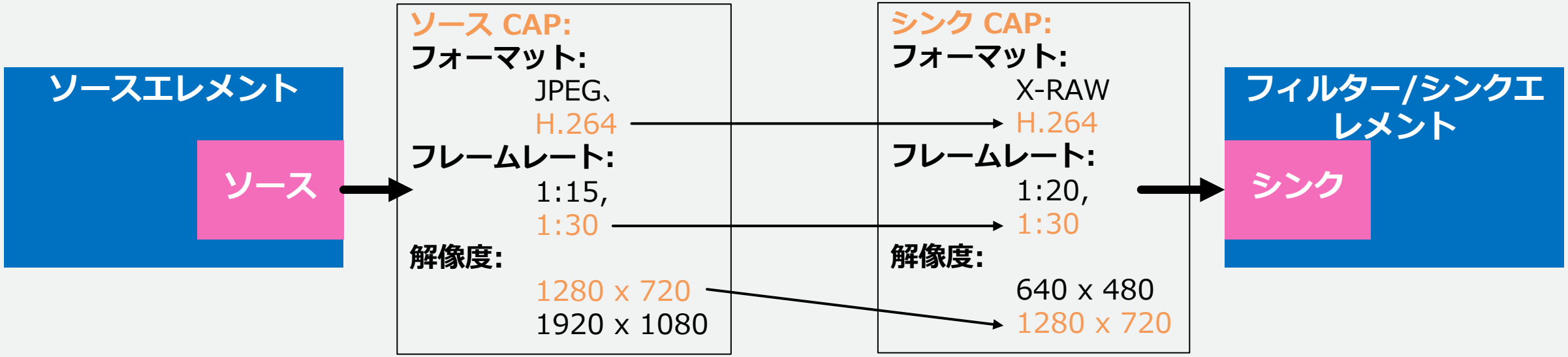
GStreamer は、ユーザーが選択した特定のメディアタイプを受け入れ(**シンク**)、必要なメディアタイプを次のエレメントに渡し(**ソース**)、エレメントをつなぎ合わせてメディアパイプラインを作成する。



https://gstreamer.freedesktop.org/documentation/application-development/basics/elements.html?gi-language=c
https://gstreamer.freedesktop.org/documentation/application-development/basics/pads.html?gi-language=c

GStreamer PAD Capabilities (or CAPs)

多くのソースパッドとシンクパッドは、フォーマット、フレームレート、解像度などの**複数の機能(CAP)をサポート**しています。GStreamer はデフォルトで、サポートされている機能を接続された**エレメント間でネゴシエート**しようとする。



対応した機能を持たないエレメント同士を接続してしまうのが、GStreamer のパイプラインエラーのよくある原因。

GStreamer CAP フィルタリング

CAP(機能)フィルターを用いると、ソースエレメントに複数の機能 (出力パラメータ) がある場合に、GStreamer エレメントに特定のメディアパラメータセットを強制的にソースさせることができる。

例:

```
$ gst-launch-1.0 avfvideosrc device-index=0 ¥
! videoconvert ¥
! video/x-raw,format=NV12,width=1280,height=720,framerate=20/1 ¥
...
```

CAP フィルターライン

CAP フィルターを**早めに、こまめに適用**することで、パイプラインのエラーや非効率な処理を回避することができる。

各段階で CAP フィルタリングを使用し、GStreamer にネゴシエートさせるのではなく、意図した属性(フォーマット、解像度、フレームレートなど)が次のエレメントに渡されるようにする。

GStreamer のデバッグ

GST_DEBUG などの環境変数を設定することで、詳細なログやエラーメッセージを確認できる。

```
$ GST_DEBUG=4 gst-launch-1.0 videotestsrc ! ...
```

出力例

```
0:00:00.066082000 95447    0x139809de0 INFO  bin gstbin.c:2768:gst_bin_do_latency_func:<pipeline0> configured latency of
0:00:00.000000000
New clock: GstSystemClock
0:00:00.077389000 95447    0x137e10d90 ERROR glcaopengllayer gstglcaopengllayer.m:161:-[GstGLCAOpenGLLayer
copyCGLContextForPixelFormat:]: failed to retrieve GStreamer GL context in CAOpenGLLayer
0:00:00.090116000 95447    0x137e10d90 INFO  glcaopengllayer gstglcaopengllayer.m:168:-[GstGLCAOpenGLLayer
copyCGLContextForPixelFormat:]: attempting to create CGLContext for CAOpenGLLayer with share context 0x138958200
0:00:00.090667000 95447    0x137e10d90 INFO  glcontext gstglcontext.c:1113:_create_context_info:<glwrappedcontext0>
GL_VERSION: 4.1 Metal - 76.3
0:00:03.840078000 95447    0x137ffe920 WARN  glimagesink gstglimagesink.c:2485:gst_glimage_sink_on_close:<sink> Output
window was closed
```

https://gstreamer.freedesktop.org/documentation/gstreamer/running.html?gi-language=c

まとめ

まとめ

- 本セミナーで紹介した Tips を活用することで Kinesis Video Streams をより良く利用できる
  - エンコードやプレイヤーの設定によって遅延を削減できる
  - 対応しているコーデックや、サービスの上限を把握しておく
  - コストファクターを理解し、ユースケースに応じて最適化する
  - WebRTC や GStreamer のよくある問題とトラブルシューティング方法を理解しておく

Thank you!