# HARK SaaS: ロボット聴覚ソフトウェア HARK のクラウドサービスの設計と開発

**HRAK SaaS: Design and Implementation of Robot Audition Software HARK as a Service**

**水本武志, 中臺一博**

**Takeshi MIZUMOTO, Kazuhiro NAKADAI**

株式会社 ホンダ・リサーチ・インスティチュート・ジャパン

**Honda Research Institute Japan, Co., Ltd.**

t.mizumoto@jp.honda-ri.com, nakadai@jp.honda-ri.com

## Abstract

本稿では、2008 年より公開を開始したロボット聴覚ソフトウェア HARK[1] をクラウドサービスとして実装した HARK SaaS (Software as a Service) について報告する。HARK SaaS は多チャネル音ファイルを受信して HARK が提供する音源定位や音源分離などの結果を返すクラウドサービスである。従来の HARK で必須であったローカル計算機へのインストール作業や、高負荷の信号処理が実行できる高い性能要求が不要となるため、より簡単に HARK を利用できる。評価実験では、Amazon Web Services (AWS) を用いてサーバ 6 台構成で応答時間と処理時間を計測した。その結果、応答時間は 100 並列アクセスまでは 100msec 程度であること、処理時間はオーバーヘッドが無視できるほど入力データが長い場合は実時間処理が可能であることを確認した。

## 1 はじめに

ロボット聴覚分野で研究開発されてきた音源定位・音源分離などのマイクロホンアレイ処理技術が実装されたロボット聴覚ソフトウェア HARK が 2008 年から公開されている [Nakadai 09]。公開以来 HARK は様々なシステム、例えばクイズの司会 [Nishimuta 15] やテレプレゼンスロボット [Mizumoto 12] に応用されている。また、ユーザビリティの面でもインストーラやドキュメントの整備を行うなど、利便性向上の継続的な努力が続けられている。しかし、既存システムへの組み込みには ROS (Robot OS) 等の別ソフトウェアやソケット通信の実装が必要となるなど、依然ハードルは高い。さらに、現在の HARK は信号処理を単一の計算機上で実行する必要があるので、計算機への要求スペックも高い。そのため、低スペックの計算機、例えば組み込みデバイスでの利用には専用の実装が必要であった [中臺 15]。

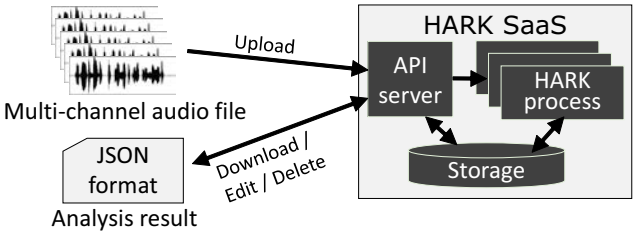


図 1: HARK SaaS の概要

一方、近年の無線ネットワーク環境の普及や Internet of Things の流行などにより、ネットワーク接続できるセンサデバイスや小型計算機が多く出回っている。例えば、Intel® Edison (Intel 社) や Raspberry Pi® (Raspberry Pi Foundation)、BeagleBoard ™ (テキサス・インスツルメンツ社、Digi-Key) に代表されるようなネットワーク接続可能な小型計算機は容易に入手できる一般的なものになっている。

これらの状況にもとづいて、本稿では、HARK をクラウドサービスとして設計・開発した **HARK SaaS** について報告する。図 1 に示す本サービスの概要のとおり、ユーザは HARK SaaS への多チャネル音ファイルアップロードと、処理結果の取得・更新・削除ができる。本サービスはサーバ側で全処理を行うため、ローカル計算機にネットワーク接続が必須となるものの、サービス利用をするだけなので従来のインストール作業を回避でき、信号処理をクラウド上のサーバへ移譲するので要求スペックが低くなる。このため、従来のローカル型 HARK の課題の解決が期待できる。

HARK SaaS 設計上の要求条件は次の 3 点である。

[1] Honda Research Institute Japan Audition for Robots with Kyoto University

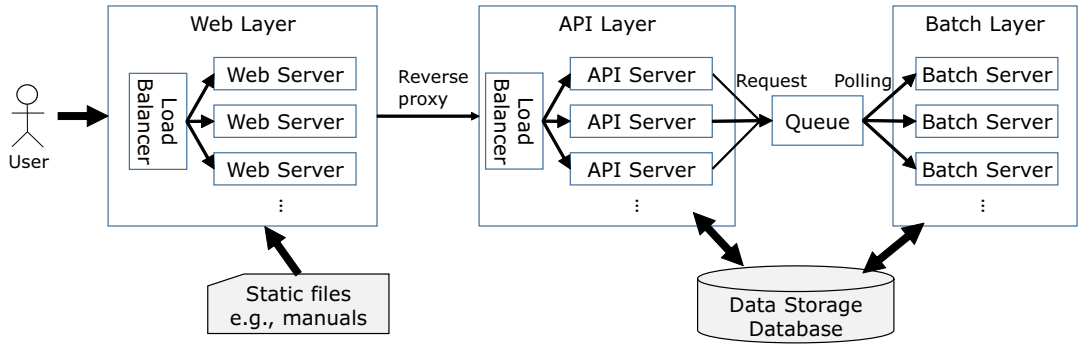


図 2: HARK SaaS のアーキテクチャ

**インタフェースの汎用性**

他のソフトウェアやクラウドサービスとの組み合わせが容易であれば、HARK SaaS の応用を行いやすくなる。そこで、標準規格に準拠した汎用的なインタフェースを持たせる必要がある。

**ユーザビリティ**

プログラムを作成せずに HARK の機能を利用したいユーザと、本サービスと組み合わせたソフトウェアを開発したいユーザの両方に利用しやすいユーザインタフェースを設計する必要がある。

**信頼性**

サービスとしてのセキュリティや安定性を高める設計が必要である。また、処理結果についても、従来のローカル型 HARK との互換性を持たせる必要がある。

本稿の構成は次のとおりである。まず、2 章で関連するクラウドサービスについて議論する。次に、3 章で本サービスのアーキテクチャやデータ構造を設計する。4 章で本サービスの基本性能に関する実験結果について議論し、5 章で本稿をまとめる。

## 2 音のクラウドサービス

本章では、音声や音楽などの音データを利用するクラウドサービス (以下、音のクラウドサービスと呼ぶ) について議論し、本サービスの位置付けを明らかにする。音のクラウドサービスには、(1) アップロードされた音データをそのまま用いるサービスと、(2) 音データの処理を伴うサービスがある。本サービスはマイクロホンアレイ処理を行うので後者に分類される。以下で述べるように様々な音のクラウドサービスが公開されているが、後者のサービスは単一チャネル処理のみであり、本サービスのように多チャネル音データを処理するサービスは筆者らの知る限り存在しない。

アップロードされた音データをそのままに用いるサービスは、ソーシャルネットワーキングや音の共有を目的としたサービスが一般的である。例えば、 SoundCloud® [2] や YouTube® [3] は、ユーザがアップロードした音データを、他のユーザと共有することができる。これらのサービスでは、音データ以外にもユーザ自身が追加したタグやコメント、5 段階評価などの音データに対する付加情報が合わせて提供される。

音データの処理を伴うサービスを、対象とするデータが音声のものと音楽のものに分類して議論する。音声を対象としたサービスには、Google 社の音声検索サービス、Apple 社のスマートフォン上で動作する音声対話サービス Siri®、ロボットインタラクションを目的とした音声認識と音声合成サービス Rospeex [杉浦 13] などがある。これらは入力された音声を認識し、認識結果そのものや、認識結果に基づく検索結果、音声応答などを返すサービスである。また、インターネット上のポッドキャストや動画を音声認識によってテキスト化し検索できるサービス PodCastle [Goto 13] は、ユーザによるアノテーションを利用した性能向上を組み合わせたサービスである。音楽を対象としたサービスには、その基礎技術となる歌声や音楽の分析・検索に関する研究が数多くされており [後藤 08]、公開されているサービスにも、SoundHound 社の ハミング検索サービス midomi®[4] や、音楽からサビ区間やメロディを推定し、表示することで能動的な音楽鑑賞を可能とする Songle [5] [Goto 11] などがある。

## 3 HARK SaaS の設計と実装

本章では、 HARK SaaS の詳細について述べる。まず 3.1 節で、 1 章で述べた 3 点の要求条件を検討しながらアーキテクチャを設計する。次に 3.2 節でデータ構造を設計し、3.3 節でサービスの実装について述べる。

### 3.1 アーキテクチャ設計

インタフェースの汎用性について検討する。まず、本サービスの全機能は HTTPS リクエストを用いること、送受

[2] https://soundcloud.com
[3] https://www.youtube.com
[4] http://www.midomi.co.jp
[5] http://songle.jp

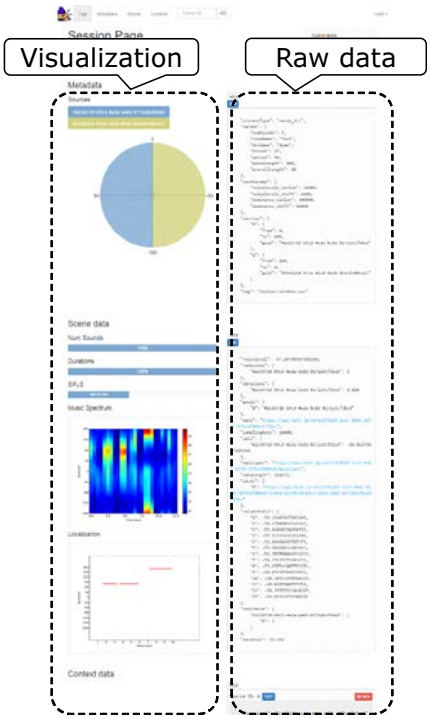


図 3: 可視化機能を備えた Web UI

```
import pyhark.saas
h = pyhark.saas.PyHarkSaaS("API_KEY", "API_SECRET")
h.login()                                # 認証
h.createSession(metadata)                # パラメータ設定
h.uploadFile(open(filename, 'rb'))       # ファイル送信
h.wait()                                 # 処理終了待ち
result = h.getResults()                  # 処理結果受信
```

図 4: HARK SaaS SDK を用いたサンプルコード

信データは JSON フォーマットを用いることとする。これらはいずれも標準規格なので、ほぼあらゆるプログラミング言語からこれらのインタフェースを介して本サービスを利用することが可能となる。次に、インタフェースの複雑さ制限するため、ローカル型 HARK の自由に信号処理のデータフローを構成できる機能に制限を加える。代わりに標準的な音源定位と音源分離を行う構成を提供し、多くのパラメータ、例えば音源定位・音源分離用伝達関数、音源定位閾値、定位長などを提供することで、インタフェースの汎用性とサービスの利便性の両立を図る。

ユーザビリティについて検討する。プログラミングを行わずに HARK の機能を利用したいユーザに対しては、Web インタフェースに解析結果の可視化機能を提供する (図 3)。本インタフェースを用いれば、ブラウザ操作のみで音ファイルの送信と結果の確認ができる。一方、プログラミングを行ってソフトウェアに組み込みたいユーザについては、Software Development Kit (SDK) を提供する。SDK は Python モジュールとして提供し、認証や HARK 処理リクエスト、結果の取得ができる。SDK を用いたサンプルコードを図 4 に示す。

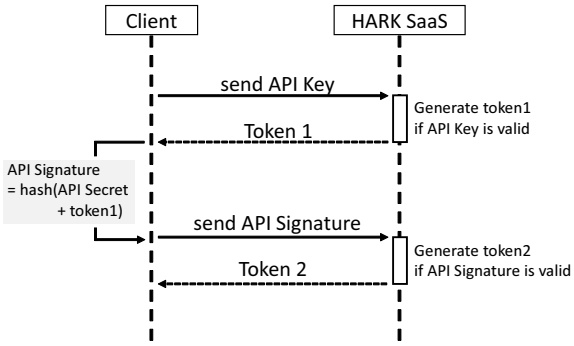


図 5: 認証シーケンス

信頼性について検討する。まず、サービスの安定性を実現するためには、処理の特性ごとにレイヤを分割しそれらを疎結合させる構造と、負荷変動に追従できるスケーラビリティを持たせる必要がある。これらを満たすアーキテクチャを図 2 に示す。前者については、(1) ドキュメントなどの静的ファイルを配信とリクエストの後段への転送を行う Web レイヤ、(2) データベースアクセスが必要なリクエストの処理と HARK 実行リクエストの後段への転送を行う API レイヤ、(3) HARK の実行や後処理などの時間のかかる処理を行う Batch レイヤに分割する。各レイヤの疎結合構造は次のように実現する。まず、Web レイヤから API レイヤへの転送には API レイヤのロードバランサを介することとする。この設計によって、Web レイヤの API レイヤサーバ台数への依存性を排除できる。次に、API レイヤから Batch レイヤへのリクエスト転送をキューを介することとする。この設計によって API レイヤのサーバと Batch レイヤのサーバはキューのみにアクセスすればよく、互いのサーバ台数への依存性がなくなる。後者のスケーラビリティについては、 Web レイヤと API レイヤの前面にロードバランサを配置する。この設計によって、負荷が高ければロードバランサに接続するサーバ台数を増やし、負荷が低ければサーバ台数を減らすことでレイヤ全体の処理性能を制御できる。

次に、サービスのセキュリティを実現するために次の設計を行う。(1) ユーザ認証。ユーザごとに 2 つの情報 (API Key, API Secret) を提供し、全てのサービスへのアクセスについて図 5 に示す手順で得られた一時認証トークン (Token2) の提供を要求する。また、一時認証トークンは短時間で無効化することで、流出時の影響を制限する。(2) 暗号化。通信を暗号化するために、本サービスへのリクエストを全て HTTPS アクセスのみに制限する。

最後に、ローカル版 HARK との互換性については、Batch レイヤのサーバでローカル版 HARK 自体を実行し、得られる結果を全て次節で設計するデータ構造で表現することとする。これによって、ローカル版と同じ実装を用いるので結果の互換性を確保できる。

### 3.2 データ構造の設計

本節では HARK SaaS で利用するデータ構造を設計する。まず、本サービスにアップロードされたひとつの音ファイルをデータ単位と定義し、**セッション** と呼ぶ。本設計では全ての処理結果や処理パラメータは全てセッション単位で表現する。

ひとつのセッションに関するデータを 3 種類に分類する。

**メタデータ**

ユーザが与えるデータ。例えば、HARK に与えるパラメータや音源方向ラベルが含まれる。音源方向ラベルとは、方向範囲ごとに定めるラベルのことで、これを適切に設定すれば、マイクロホンアレイと音源の位置関係が変化しない場合 (会議など) に、音源定位された音イベントへラベルを自動付与できる。

**コンテキスト情報**

音イベント毎のデータ。HARK によって音源定位された音イベント毎に定義される。例えば、音イベントの開始時間と終了時間、仰角と方位角、音量、分離音などが含まれる。

**シーン情報**

シーン全体のデータ。コンテキスト情報を集計した結果など、ひとつのセッション全体に対して定義される。例えば、処理される音ファイルの長さやサンプリングレート、音量時系列等の音ファイルそのものに関する情報や、音源方向ラベルごとの音イベント数、その合計時間などの音源方向ラベルごとの情報が含まれる。

### 3.3 HARK SaaS 実装

本サービスを AWS 上に実装した。各コンポーネントに用いた AWS のサービスは次のとおりである。Web レイヤと API レイヤの負荷分散には Elastic Load Balancer (Amazon ELB) を、Batch レイヤが監視するキューには Simple Queue Service (Amazon SQS) を、アップロードされる音声ファイルや処理結果の保存には Simple Storage Service (Amazon S3) を、処理結果などのその他のデータの保存は Amazon RDS を利用した。

## 4 評価実験

本章では、HARK SaaS の評価実験について述べる。実験では、応答時間と処理時間の計測を通して本サービスの基本性能を明らかにし、アプリケーションのサンプルによって本サービスの応用例を示す。

### 4.1 実験設定

実験に使用した HARK SaaS サービスの構成は 3 章で述べたとおりである。実験では、 Web, API, Batch 各レイヤに、サーバを 2 台ずつ割り当てた。また、全ての実験でローカル計算機は 1 台のみを使用し、大規模な負荷試験で標準的に行われる複数台の計算機を用いたアクセスは行っていない。実験に使用した音ファイルには、標準的な室内で 8 チャネルのマイクロホンアレイで収録した 6 名の自由会話を用いた。処理時間計測で用いる音ファイルは 10 秒, 10 分, 30 分のデータとし、応答時間計測で用いる音ファイルはすべて 10 秒のデータとした。

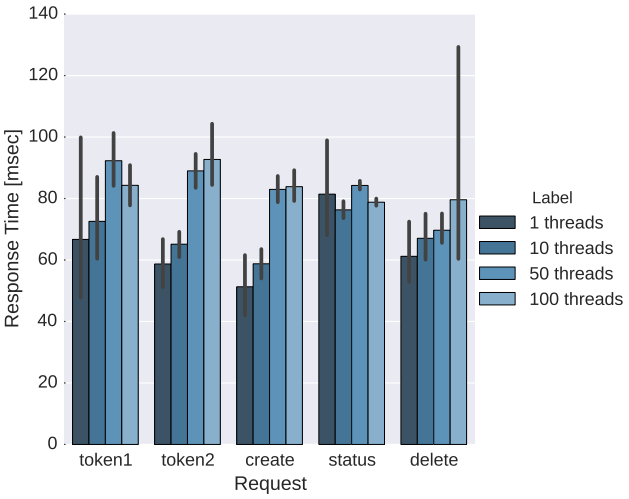


図 6: 実験 1: リクエストの応答速度

### 4.2 実験 1: 応答時間

応答時間計測には、 オープンソース・ソフトウェア JMeter [6] を利用した。試験シナリオは、認証から処理リクエスト、結果の取得と削除までの一連の処理とした。本シナリオは次の 6 種類のリクエストで構成される。

1. Token 1 (認証)
2. Token 2 (認証)
3. Create (セッションの作成)
4. Upload (データアップロード)
5. Status (処理状態確認)
6. Delete (セッション削除)

上記シナリオを、同時並列実行数を 1, 10, 50, 100 と変化させながら各 10 回ずつ実行し、応答時間を計測した。

応答時間を図 6 と 図 7 に示す。横軸のラベルは試験シナリオの各リクエストを表し、ラベル中の棒グラフはそれぞれ同時アクセス数に対応した応答時間を表し、エラーバーは当該リクエストの応答時間の標準偏差を表す。

図 6 に示す通り、 100 並列アクセスの場合でも応答時間は 100msec 程度を維持しているので、この負荷であれば安定した処理ができているといえる。一方で、ファイルアップロードについては 図 7 に示す通り、同一のファイルをアップロードしたにもかかわらず応答時間が伸びて

[6] http://jmeter.apache.org/

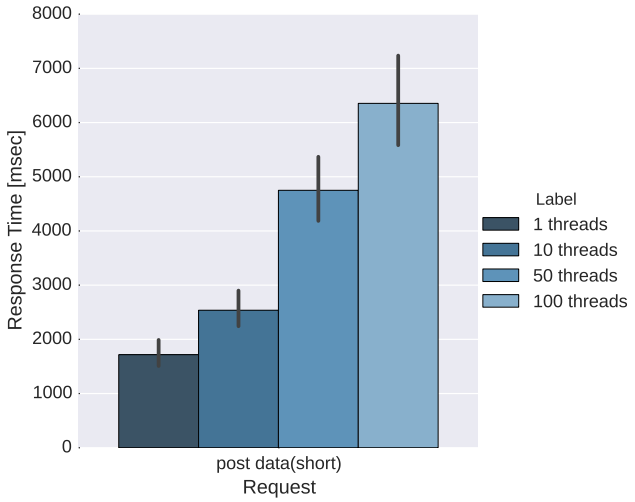


図 7: 実験 1: ファイルアップロードリクエストの応答速度

いる。つまり、本実験で用いた 2 台の構成では処理が間に合わず、待ちが発生している。

これより、本実験で用いた構成の性能では、結果取得や認証の処理には足るものの、同様の規模でファイルのアップロードの処理には足りない。したがって、アップロード量の状況に応じた台数の増減が必要である。

### 4.3 実験 2: 処理時間

ローカル版 HARK では実時間で音ファイルの処理を行えるが、HARK SaaS ではリクエストの処理やデータ転送等のオーバーヘッドが含まれるので、システム全体の処理時間はより長くなる。本実験では, 異なるデータ長の多チャネル音データの処理時間を計測し、HARK SaaS 全体としての処理速度を評価する。実験には、8 チャネルの音ファイルを使用し、その長さは 3 種類 (10 秒, 10 分, 30 分) とした。また、音源定位のされやすさに関する閾値を 26 - 30 まで 1 ずつ 5 段階に変化させ、音源定位数による処理時間の変化も評価した。

実験結果を図 8 に示す。縦軸が HARK SaaS へのリクエストが受理されてから処理結果が戻るまでの全体の処理時間を表し、横軸は入力データ長を表す。また、処理時間と入力データ長の比で表されるリアルタイムファクタ (RT) を表 1 に示す。RT とは実時間制を表す指標で、 1 より小さければ、入力データ長より処理時間が短いので、実時間性があると言える。

結果について議論する。まず、閾値を変化させてもほぼ処理時間に変化はなかった。これは、HARK 処理と音イベントの後処理を並行して行っているために後処理の時間の影響が小さいことが理由であると考えられる。次に、表 1 に示すとおり 10 分以上のデータでは実時間性を確保できているが、10 秒では確保できていない。これは、クラウドサービス化に伴うオーバーヘッドの影響の方が、HARK の処理時間の短さよりも大きいことが原因である。

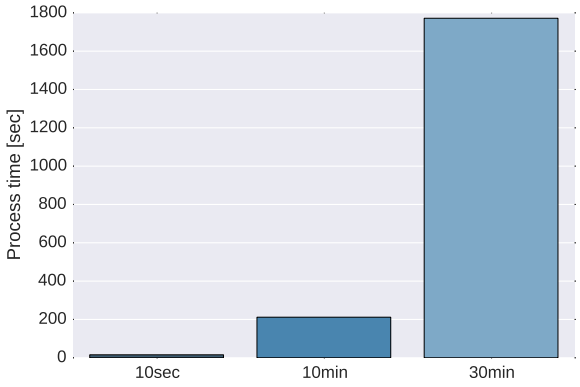


図 8: 実験 2: 入力データごとの処理時間計測結果

表 1: 実験 2: リアルタイムファクタ

| データ長 [sec] | 処理時間 [sec] | RT |
|---|---|---|
| 10 | 15.6 | 1.56 |
| 300 | 211.7 | 0.35 |
| 1800 | 1771.7 | 0.98 |

### 4.4 実験 3: HARK SaaS サンプルアプリケーション

HARK SaaS の応用システムの例として、音環境可視化アプリケーションを構築した (図 9 ) 本アプリケーションは、録音された音ファイルを HARK SaaS へアップロードし、処理結果を受信し、処理結果とメタデータで設定された音源方向ラベルを利用して結果を可視化する。本アプリケーションは音源方向ラベルごとの音イベント集計結果を色分けして表示するので、方向ごとの音環境分析ができる。なお、本アプリケーションは HARK SaaS の Python SDK と、可視化ライブラリ Seaborn、 Matplotlib を使用している。

可視化画面は 4 部分から構成されている。まず、図左上は方向ごとの音イベント数を表し、音源方向ラベルごとに色分けがされている。この図から、音源方向ラベルの方向に関する傾向が分かる。例えば図 9 の場合、緑色の音源方向ラベルの音イベントは 0 度 方向から多く発生していることがわかる。次に、図右上部は音源方向ラベルごとの音イベントの継続時間のヒストグラムと平均値を表す。ヒストグラムから音源方向ラベルごとの継続時間の傾向を分析でき、右端の平均値から音源方向ラベル同士の継続時間の比較ができる。続いて、図右中部は音源方向ラベルごとの音イベント音量のヒストグラムと平均値を表す。ここでも継続時間と同様に音源方向ラベルごとの分析やそれぞれの比較ができる。最後に、図下部は時間・方向ごとの音イベントを表す。この図より、-120 度の方向からは 100 秒から 180 秒、280 から 380 秒、 430 秒から 500 秒の 3 回にわたって音イベントが連続的に発生していることがわかる (濃青で表示)。

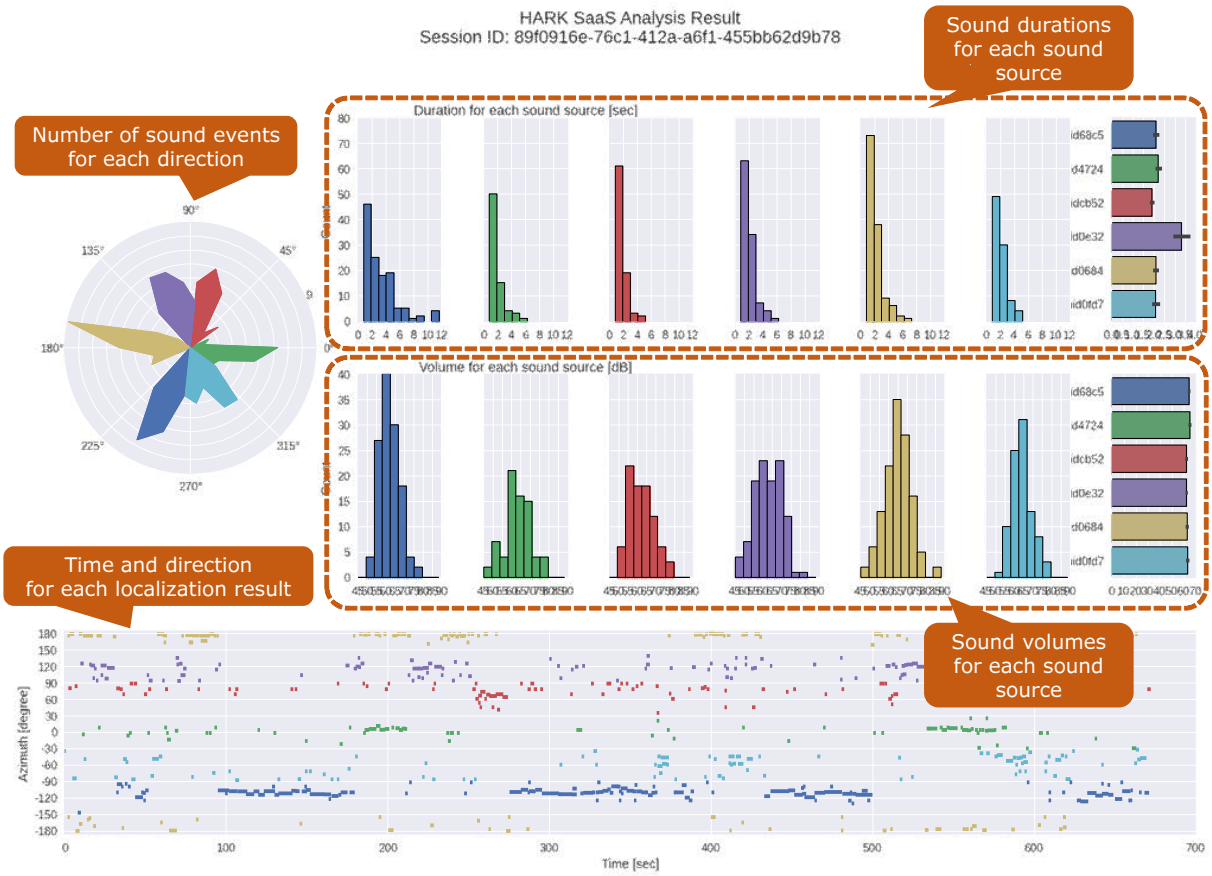


図 9: HARK SaaS サンプルアプリケーション： 音環境可視化

## 5 まとめ

本稿では、ロボット聴覚ソフトウェア HARK をクラウドサービスとして実装した HARK SaaS について報告した。実験の結果、100 並列アクセスまでは応答時間が 100msec 程度と安定していること、10 秒のデータの場合はオーバーヘッドのために実時間性が失われるものの、それより長いデータであれば実時間性が確保できることが明らかになった。今後は、応答速度の向上やより大規模なアクセスに耐えうる冗長設計、分離音の後処理の充実による機能拡大などを行う予定である。

## 参考文献

[Goto 11] Goto, M., Yoshii, K., Fujihara, H., Mauch, M., and Nakano, T.: Songle: A Web Service for Active Music Listening Improved by User Contributions, in *ISMIR*, pp. 311–316 (2011)

[Goto 13] Goto, M., Ogata, J., and Eto, K.: PodCastle: A web 2.0 Approach to Spech Recognition Research, in *Interspeech*, pp. 2397–2400 (2013)

[Mizumoto 12] Mizumoto, T., Nakadai, K., Yoshida, T., R. Takeda, T. T., T. Otsuka, and Okuno, H. G.: Design and Implementation of Selectable Sound Separation on the Texai Telepresence System using HARK, in *ICRA*, pp. 694–699 (2012)

[Nakadai 09] Nakadai, K., Okuno, H. G., Nakajima, H., Hasegawa, Y., and Tsujino, H.: Design and Implementation of Robot Audition System “HARK”, *Advanced Robotics*, Vol. 24, pp. 739–761 (2009), doi:10.1163/016918610X493561

[Nishimuta 15] Nishimuta, I., Yoshii, K., Itoyama, K., and Okuno, H. G.: Toward a Quizmaster Robot for Speech-based Multiparty Interaction, *Advanced Robotics*, Vol. 29, No. 18, pp. 1205–1219 (2015)

[後藤 08] 後藤 真孝, 齋藤 毅, 中野 倫靖, 藤原 弘将：歌声情報処理の最近の研究, 日本音響学会誌, Vol. 64, No. 10, pp. 616–623 (2008)

[杉浦 13] 杉浦 孔明, 堀 智織, 是津 耕司：rospeex:クラウド型音声コミュニケーションを実現する ROS 向けツールキット, 電子情報通信学会技術研究報告, クラウドネットワークロボット, 第 113 巻, pp. 7–10 (2013)

[中臺 15] 中臺 一博, 水本 武志, 中村 圭佑：モバイル端末用マクロホンアレイシステムの開発とコミュニケーション支援への適用, ロボット学会学術講演会 (2015)