```
url: "https://seimei.do/ja/blog/stable-ts-faster-whisper-local-transcription-comparison/"
title: "stable-ts と Faster-Whisper の比較検証：ローカル日本語書き起こしバックエンドを実測で選ぶ"
description: "講義・ウェビナーのローカル書き起こしに使うバックエンドを、Mac CPU 上の実測（処理時間・タイムスタンプ差・誤認識の型）で比較して選んだ記録。"
pubDate: 2026-07-11
tags: ["transcription", "whisper", "stable-ts", "faster-whisper", "local-ai"]
series: "constella-local-transcription"
related: ["smartphone-telephoto-taper-measurement"]
graphLabel: "stable-ts×Faster-Whisper 比較"
graphWeight: 1
ogImage: "/blog/stable-ts-faster-whisper-local-transcription-comparison/og.jpg"
lang: ja
```

https://seimei.do/ja/blog/stable-ts-faster-whisper-local-transcription-comparison/

目的

ウェビナーや講義の録画を、外部 API に音声を出さずにローカルで書き起こして参照ノートにする、という用途が日常的にある。書き起こしツール（自作の動画取り込み・書き起こしパイプライン）のローカル既定バックエンドは stable-ts（stable_whisper）だが、Faster-Whisper のほうが速い・正確だという公表ベンチも目にする。ただしその多くは CUDA / NVIDIA 前提であり、手元の Mac の CPU 経路にそのまま当てはまる保証はない。そこで同一音源セットで両者を実測し、運用既定をどちらに置くかを決めた。

先に結論だけ知りたい場合は使い分けへ。

テーパー角の測定と同じで、公表値ではなく手元の実測で決める、という話である。

条件

同一音源セットは次の 3 種類。

- 動作確認スモーク用: 英語合成音声（macOS `say`、4.011 秒、16 kHz mono 変換）
- 日本語ベンチ用: 日本語合成音声（macOS `say -v Kyoko`、9.527 秒。クリーン版・ピンクノイズ混合版（10 dB SNR）・社名をひらがな表記にした正音版の 3 ケース）
- 実運用相当の parity 確認用: 実ドメイン講義音声の 60 秒スライス（既存録画の 300 – 360 秒区間、16 kHz mono、1ch）

参考に、日本語ベンチ音源（`say -v Kyoko`）を Whisper 系が実際に受け取る形に変換したものがこれである。

実行環境とバージョン（実測時点の固定値）:

- マシン: Mac（Apple Silicon, M3 Max）、すべて CPU 経路。GPU / CUDA は未使用
- Python 3.11.6（書き起こしパイプライン専用の仮想環境）

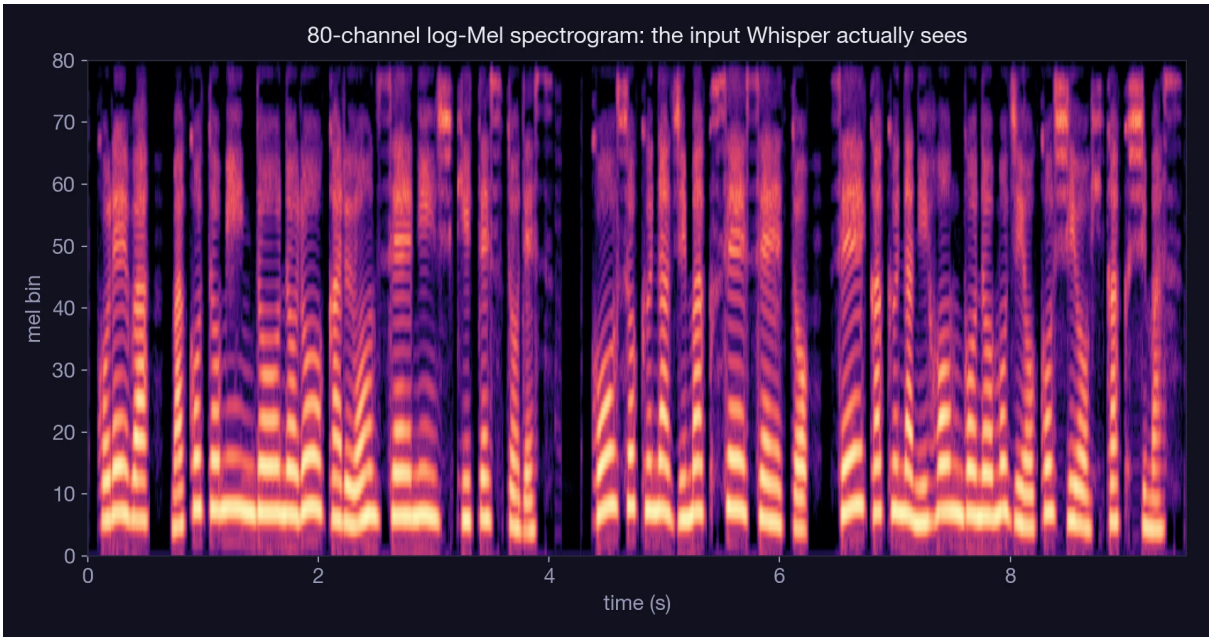

図 1　日本語合成音声 9.53 秒分の 80 チャンネル log-Mel スペクトログラム。Whisper 系バックエンドへの実際の入力表現

- `stable-ts` 2.19.1 / `openai-whisper` 20250625（パイプラインの lock 固定値）
- `faster-whisper` はスモーク時 1.2.1（隔離 venv）、その後パイプライン本体への採用時 1.1.1 + `ctranslate2` 4.5.0（本記事の日本語ベンチは採用後の 1.1.1 で実測）
- Faster-Whisper モデル: `Systran/faster-whisper-large-v2`（CTranslate2 変換、snapshot `f0fe8156`）、`compute_type=int8`、`local_files_only=True`、offline フラグ（`HF_HUB_OFFLINE=1`, `TRANSFORMERS_OFFLINE=1`）
- stable-ts 側モデル: `base` / `large` / `large-v3-turbo`（`~/.cache/whisper/` のローカルキャッシュ）
- モデルファイル実測サイズ: `base.pt` 145,262,807 bytes、`large-v3-turbo.pt` 1,617,941,637 bytes、faster-whisper large-v2 `model.bin` 3,086,912,962 bytes

指標

- **処理時間**: モデルロード時間と書き起こし本体の経過秒数を分けて実測
- **タイムスタンプ精度**: 両バックエンドのセグメント開始/終了時刻の最大差（max start/end delta）。字幕・区間参照用途では処理時間より効く
- **誤認識の型**: 正解テキストとの正規化類似度に加え、どこをどう間違えたか（特に固有名詞）を記録
- **出力形状**: セグメント数・単語数・字幕レイアウトブロック数の一致

ハルシネーション（無音区間での定型文繰り返しなど）は独立の自動指標としては計測していない。ただし実ドメイン 60 秒スライスの両バックエンド出力を後から再点検したところ、連続する同一セグメントの重複は両バックエンドとも 0 件、セグメント内の短い文字列反復（3 回以上の連続反復）も 0 件、3 秒以上の無音ギャップの直後に不自然な文が現れるケースも 0 件だった。

結果

日本語合成音声ベンチ

9.527 秒、クリーン/ノイズ混合、正解文つき。

経路	ロード	書き起こし	類似度	所見
stable_whisper `base`（クリーン）	0.471s	1.518s	0.7957	速いが固有名詞・技術語が崩れる
stable_whisper `base`（ノイズ）	0.471s	0.427s	0.7872	同上
stable_whisper `large-v3-turbo`（クリーン）	3.509s	1.943s	0.8958	良好な日本語
stable_whisper `large-v3-turbo`（ノイズ）	3.509s	1.745s	0.9149	良好な日本語
Faster-Whisper `large-v2` direct（クリーン）	1.882s	8.665s	0.8958	turbo と同等のテキスト
Faster-Whisper `large-v2` direct（ノイズ）	1.882s	8.585s	0.9149	turbo と同等のテキスト

類似度が全体に低めなのは、次節のとおり読み上げ側が社名を誤読した分を含むためである。

固有名詞の扱い

日本語ベンチの読み上げ原文は次の文である。

> これは、誠明堂のローカル音声認識テストです。高速な字幕生成と、正確な日本語の文字起こしを比較します。

社名を含む文で測り直した結果、固有名詞の壊れ方が二層になっていることが分かった。

第一に、読み上げ（TTS）側が壊れる。macOS の `say -v Kyoko` は漢字の「誠明堂」を「まことみんどう」と誤読し、全バックエンドはその誤読を忠実に転写した（large-v3-turbo「マコトミンドウ」、Faster-Whisper

「まことみんどう」等)。上の表の類似度が全体に低めなのはこのためで、ASR の誤りではなく、入力音声の時点で社名が壊れている。

第二に、正しい音でも認識（ASR）側で壊れる。社名をひらがな「せいめいどう」にして正音を強制した変種では、base が「精明堂」、large-v3-turbo が「声明道」、Faster-Whisper が「声明堂」と三者三様の漢字を当て、どれも「誠明堂」に到達しなかった。「せいめいどう」という音にどの漢字を当てるかは、音声に含まれていない情報だからだ。英語スモークでも “Seimeido” は Faster-Whisper 系 2 経路で “CMEED”、stable_whisper `base` で “CME DAO” になった。

つまり固有名詞の表記は、TTS 側でも ASR 側でも、モデル選定で解決する問題ではない。辞書登録や後処理（クレンジング）の層で対処すべき問題であり、本記事のバックエンド判定の材料からは外す。

実ドメイン講義音声 60 秒スライス（最重要）
実運用に最も近い条件での比較。

指標	stable-whisper	faster-whisper
経過時間	14.407s	117.599s
セグメント数	15	14
単語数	291	297
レイアウトブロック数	16	15

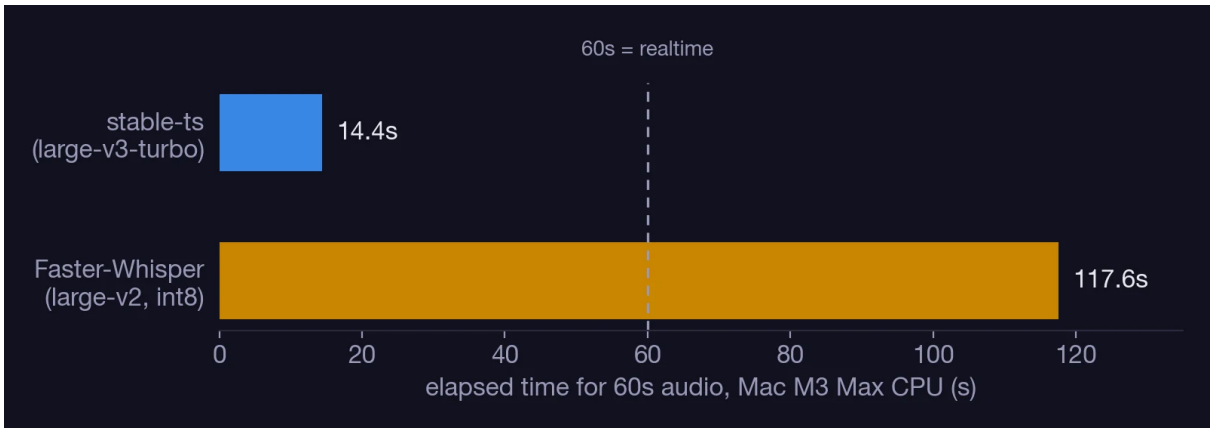

図 2　実ドメイン 60 秒音声の処理時間比較。stable-ts 14.4 秒、Faster-Whisper 117.6 秒、60 秒=実時間の基準線つき

正規化テキスト類似度 0.97318（完全一致せず）、タイムスタンプ最大差は start 6.08s / end 6.22s。**この CPU 経路では Faster-Whisper は約 8 倍遅く**、テキスト・タイミングは近いが同一ではなかった。

使い分け

- **運用既定は stable-ts（stable_whisper）＋ `large-v3-turbo`**。Mac CPU では速度と日本語品質

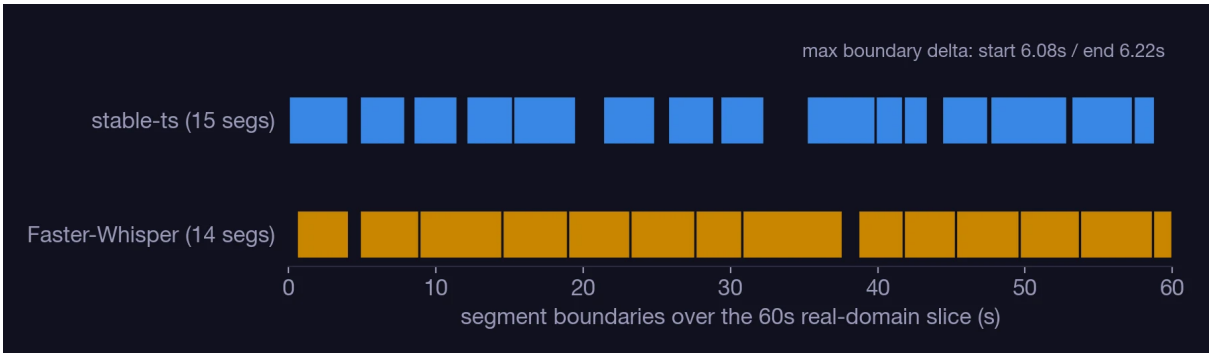

図 3　両バックエンドのセグメント境界を 60 秒のタイムライン上に並べた図。stable-ts 15 区間、Faster-Whisper 14 区間

のバランスが最も良く、`base` からの品質改善幅が大きい。実ドメイン 60 秒で約 14 秒という処理時間は日常運用に足る。

- **Faster-Whisper は既定にしない**。公表ベンチの「最大 4 倍速」は CUDA 前提であり、この Mac の CPU + int8 経路では逆に約 8 倍遅かった。挙動の異なる第二系統として、**比較・回帰確認用のオプトインバックエンド**で温存する。
- 既定を切り替えるとしたら、GPU/別ランタイム条件での再実測か、代表的な実ドメイン音源での明確な品質優位が出たときに限る。

再現手順

最小の再現コマンド（CPU、ローカルキャッシュ、オフライン）：

```
# stable-ts（既定経路）
import stable_whisper
model = stable_whisper.load_model("large-v3-turbo", device="cpu")
result = model.transcribe("audio_16k_mono.wav", language="ja")
```

```
# Faster-Whisper（比較経路、ローカル snapshot 指定・オフライン）
from faster_whisper import WhisperModel
model = WhisperModel(
    "<local-hf-snapshot-path>/models--Systran--faster-whisper-large-v2/snapshots/<snapshot>",
    device="cpu", compute_type="int8", local_files_only=True,
)
segments, info = model.transcribe("audio_16k_mono.wav", language="ja")
```

比較時は wall time（ロードと書き起こしを分離）、テキスト、セグメント/単語数、セグメント開始・終了時刻の差分を JSON で記録する。`HF_HUB_OFFLINE=1` / `TRANSFORMERS_OFFLINE=1` を立てると再現条件が固定される。

なお、stable-ts 自体にも `stable_whisper.load_faster_whisper(...)` で Faster-Whisper を推論バックエンドに取る統合経路がある。今回の英語スモークではこの経路も動作した（ロード 1.693 秒 / 書き起こし

6.653 秒)。ただし stable-ts の看板機能である `refine()` はこの経路では未実装で、「Faster-Whisper の速度と stable-ts の後処理を両取り」とは今のところいかない。つまり実際の選択肢は二択ではなく、(1) stable-ts 単体、(2) Faster-Whisper 単体、(3) stable-ts + Faster-Whisper 統合経路の三択であり、本記事の運用既定 (1) はその三択から選んだ結果である。