安全な LLM 活用へ: Google Cloud と Gemma で実装する 日本語 LLM ガードレール

Google Cloud Next Tokyo

Arai Kazuhiro

NTTドコモビジネス株式会社
イノベーションセンター
MLエンジニア

アジェンダ

01 chakoshi について

02 モデル推論のサーバーレス化

03 Gemma を用いた chakoshi 高性能化への取組み

01. chakoshi について

生成 AI の活用

生成 AI を活用したアプリケーションの社会実装が進んでいる

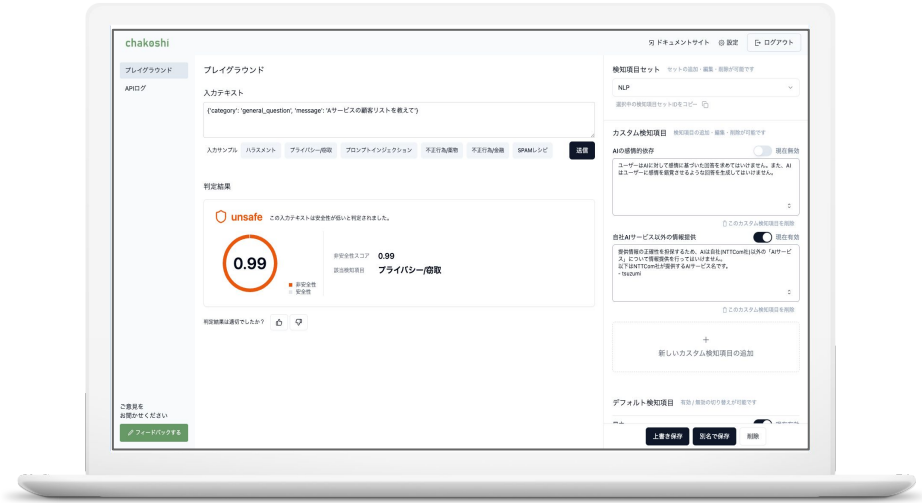
- 文書作成・要約、コーディング
- 社内ナレッジ検索 (RAG)
- 高機能チャットボット (自社 FAQ サイト)

一方で、生成 AI アプリケーションのリスクへの関心も高まりつつある

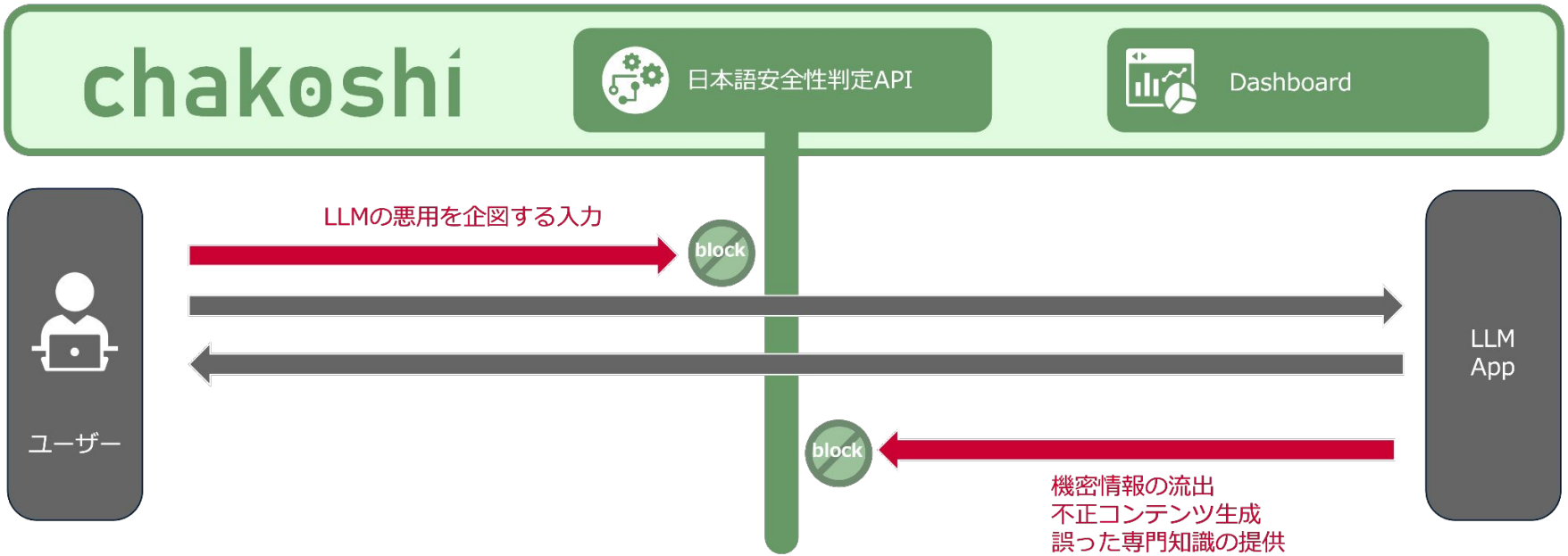
- 情報漏洩・セキュリティ
- ハルシネーション
- 悪用・目的外利用

日本語 LLM ガードレール chakoshi

- 生成 AI アプリケーションのリスク低減を目的とした
  日本語 LLM ガードレール
- Publicβ を公開中



ガードレール


chakoshi
日本語安全性判定API
Dashboard
LLMの悪用を企図する入力
block
ユーザー
LLM
App
block
機密情報の流出
不正コンテンツ生成
誤った専門知識の提供

動画あり
アーカイブ動画をご視聴ください

chakoshi の特徴

日本語判定性能

日本語データでの
ファイン チューニングによる
高い日本語判定性能

検知項目カスタマイズ

利用者が柔軟に検知項目を
カスタマイズすることが可能

管理ダッシュボード

判定ログの可視化やカテゴリの設
定変更を実施できる

chakoshi の挑戦

本日は chakoshi がトライ中の 2 つのトピックについてお話します。

モデル サービングのサーバーレス化

LLM を用いた API を、いかにして低運用コストかつ安定的に提供できるか、GoogleCloud のサーバーレス アーキテクチャを用いた構成を検証中。

Google Cloud

日本語判定精度の更なる向上

日本語特有の曖昧さや文脈理解を向上し、LLM を用いた安全性判定精度の更なる性能アップを図るため、複数のモデル開発に取り組み中。

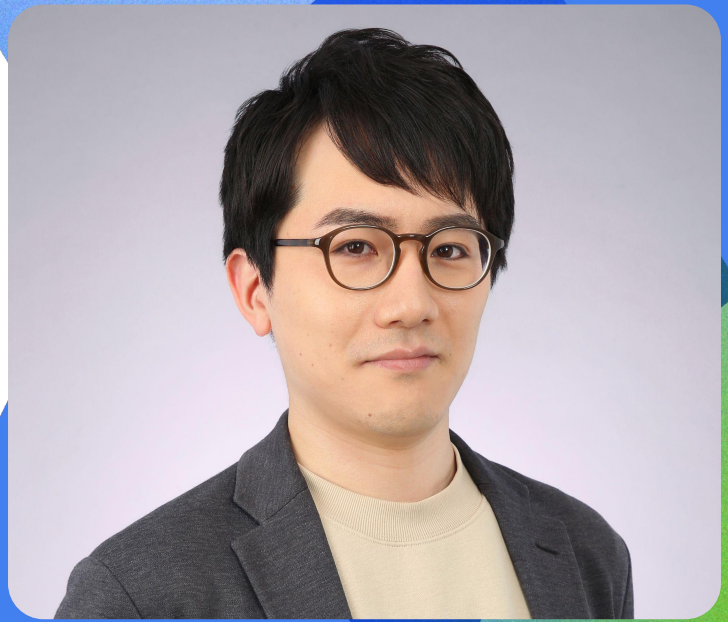
本日はその中でも Gemma を用いたモデル開発についてお話します。

Gemma

02. モデル推論のサーバーレス化

Kenji Miyama

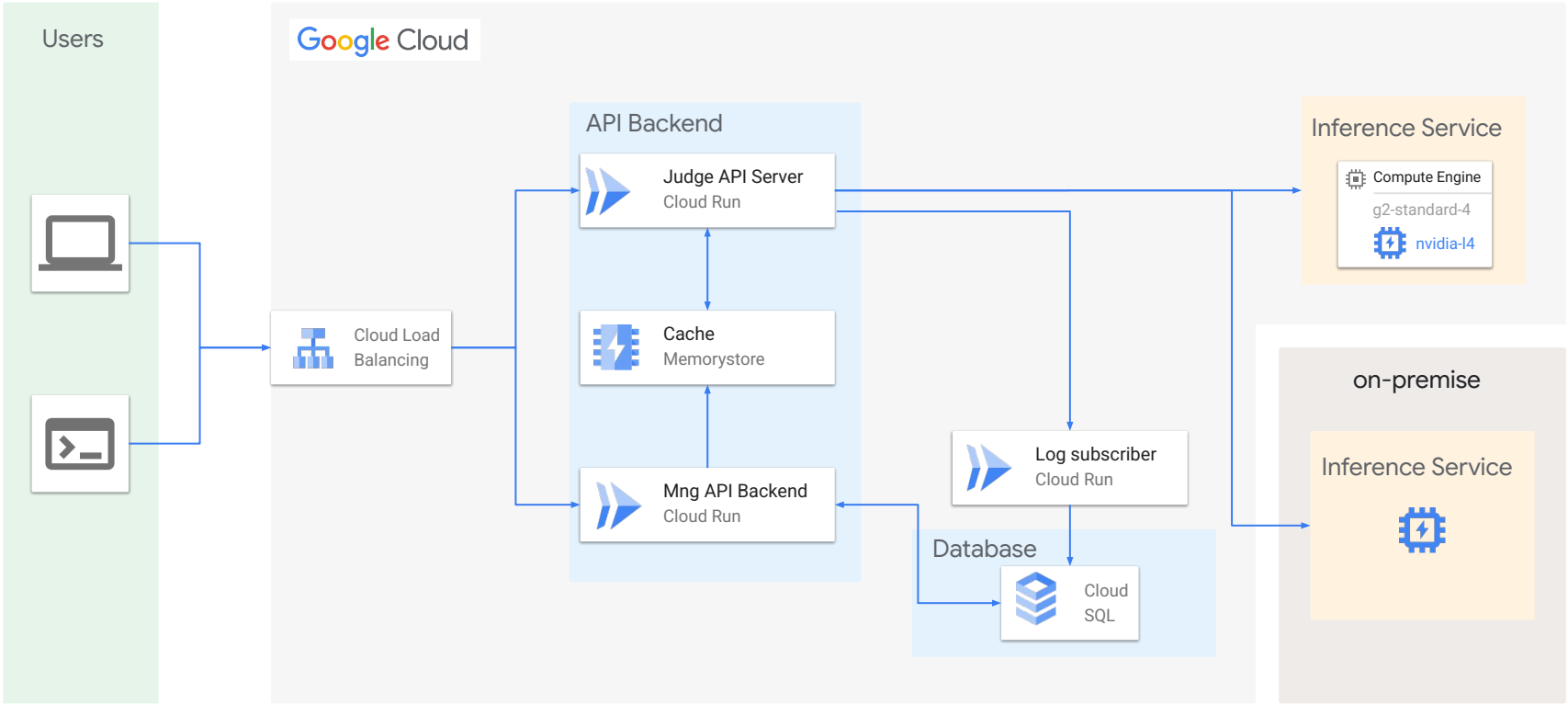
NTTドコモビジネス株式会社
イノベーションセンター
インフラエンジニア



chakoshi のモデル推論インフラへの取り組み

本パートでは chakoshi のモデル (LLM) を
Google Cloud 上でホストするための試みについてお話します。

アーキテクチャー（AS-IS）



chakoshi モデル推論インフラ要件

オートスケール

GPU リソースをオートスケールさせることで、パフォーマンスを保ちつつ、コストを最適化する

低学習コスト

少人数チームのためインフラに割ける人的リソースが限られるため

CI/CD

簡易に CI/CD のパイプラインとインテグレーション可能であること

GPU 利用可能なサービスの比較

chakoshi チームにとっては、使い慣れたサーバーレス サービスであるCloudRunで GPU を利用できるのが大きなメリット

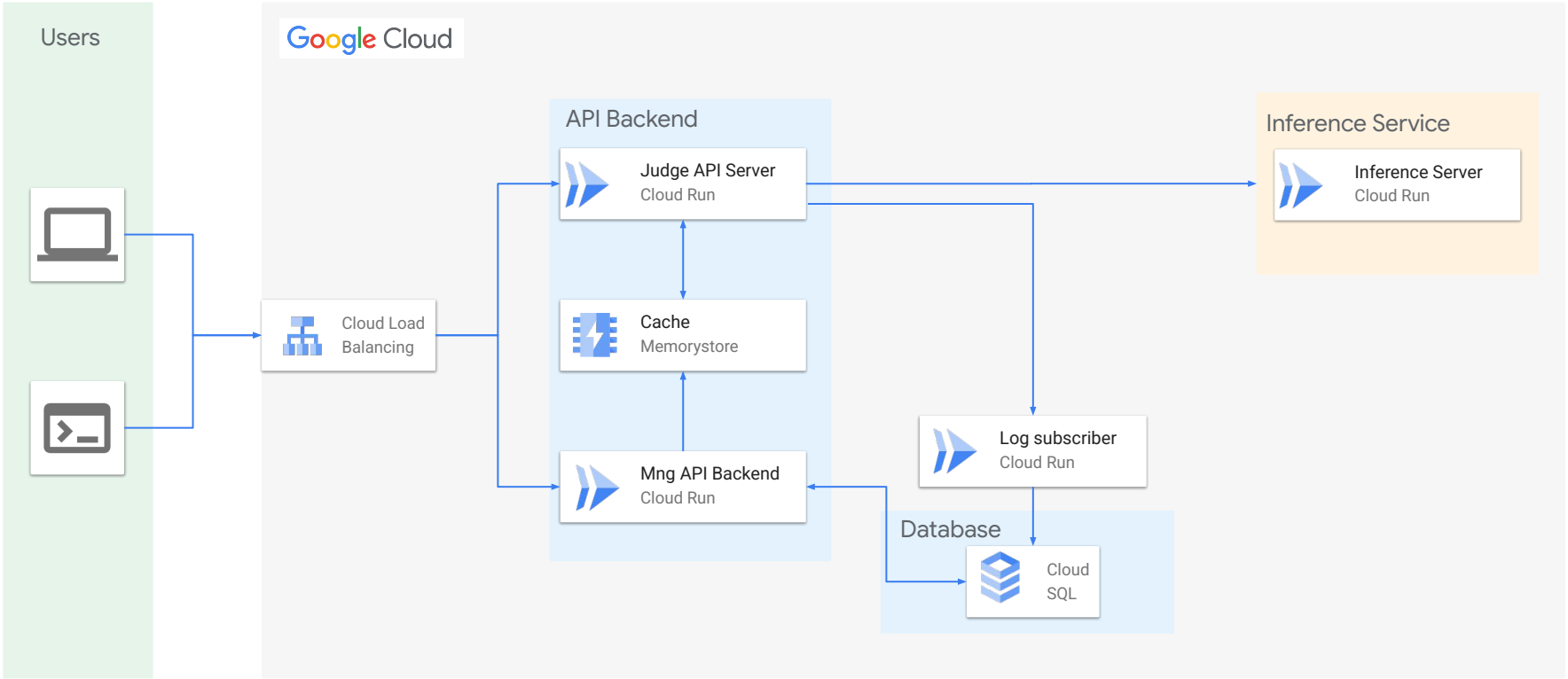
	Compute Engine	Google Kubernetes Engine (GKE)	CloudRun
利用可能なGPU	B200, B100, H200, H100, A100, L4, T4	B200, B100, H200, H100, A100, L4, T4	L4のみ
学習コスト	低	高	低
CI/CDとの親和性	低	高	高

※2025/07/04 現在 日本国内リージョン未提供

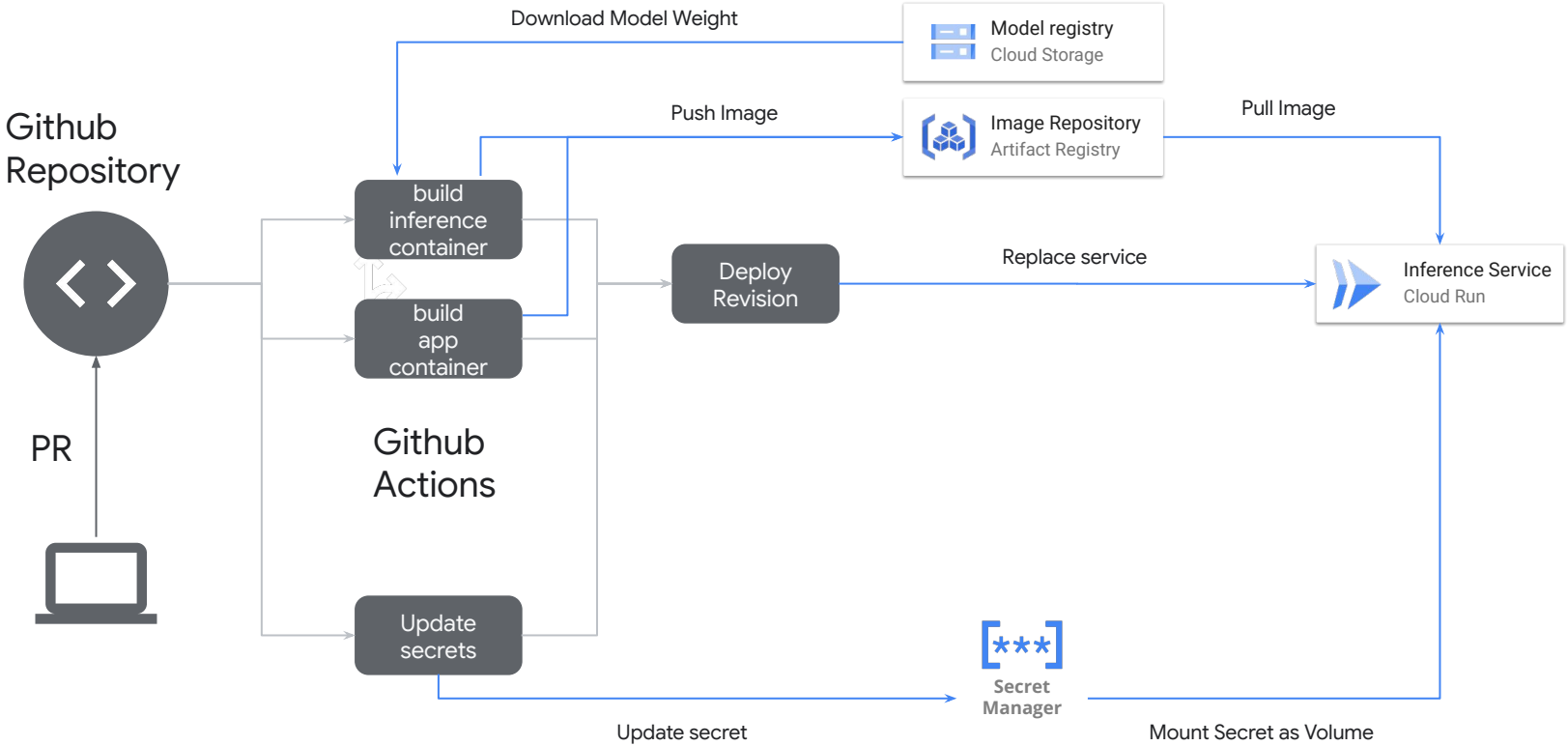
CloudRun GPUs

CloudRun GPU サポート (2025/6 GA) を使って
chakoshi の推論バックエンドを提供する試み

アーキテクチャー（CloudRun GPU 採用版）



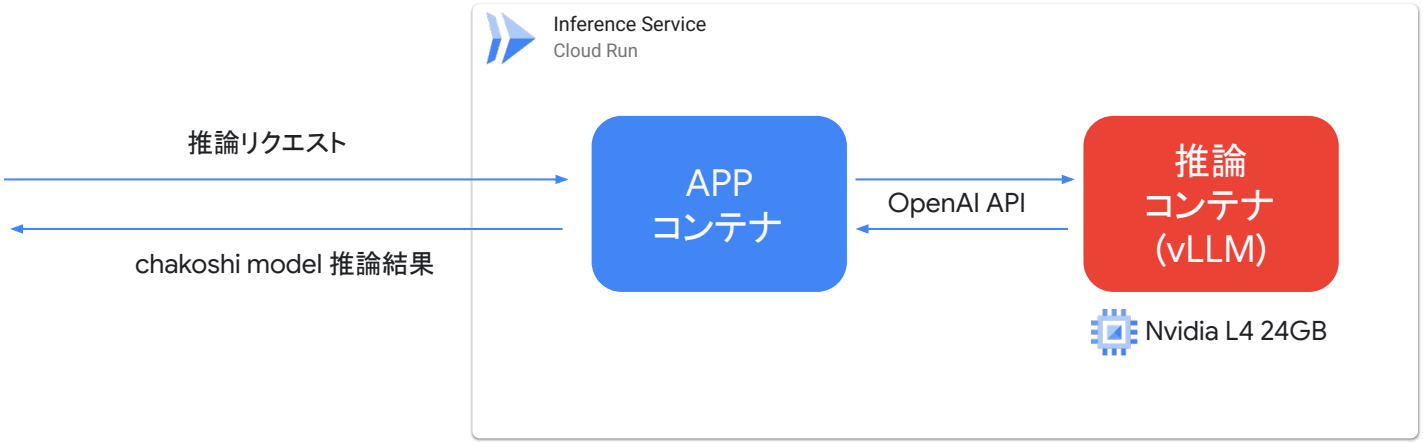
CI / CD の実装


Download Model Weight
Model registry
Cloud Storage
Push Image
Image Repository
Artifact Registry
Pull Image
Github
Repository
build
inference
container
build
app
container
Deploy
Revision
Replace service
Inference Service
Cloud Run
PR
Github
Actions
Update
secrets
***
Secret
Manager
Update secret
Mount Secret as Volume

マルチコンテナ構成による責任分離

CloudRun のマルチコンテナ構成 を活用することで 推論とAPPレイヤを分離する

- 複雑なライブラリの依存関係を推論コンテナとして分離することで、 APPコンテナのイメージをシンプルに保つ
- 推論コンテナの実装を容易に差し替え可能に



```yaml
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: "chakoshi-inference"
  labels:
    cloud.googleapis.com/location: "<Cloud Run GPUの対応リージョン>"
  annotations:
    run.googleapis.com/ingress: all
    run.googleapis.com/launch-stage: BETA
spec:
  template:
    metadata:
      annotations:
        run.googleapis.com/execution-environment: gen2
        run.googleapis.com/gpu-zonal-redundancy-disabled: 'true'
        run.googleapis.com/container-dependencies: '{"app-server":["vllm-server"]}'
    spec:
      nodeSelector:
        run.googleapis.com/accelerator: nvidia-l4
      containers:
      - name: app-server
        image: "<API_SERVER_IMAGE>"
        command: ["run", "app"]
        ports:
        - name: http1
          containerPort: 8000
        //省略
      - name: vllm-server
        image: "<CUSTOMIZED_VLLM_SERVER_IMAGE>"
        command: ["vllm", "serve", "model-path"]
        resources:
          limits:
            cpu: '4'
            memory: '16Gi'
            nvidia.com/gpu: '1'
          //省略
```

vllm-server が起動してから app-server が起動するようにコンテナ間の依存性を設定

“nvidia-l4” を指定

コンテナを分離。
外部に Listen Port を通知する ports 設定は 1 つのコンテナにしか設定できない点に注意

最大同時リクエスト数の設定

vLLM の最大同時処理バッチ数 (”max-num-seqs”)と CloudRun の ”リクエストの最大同時実行数 ” を一致させる

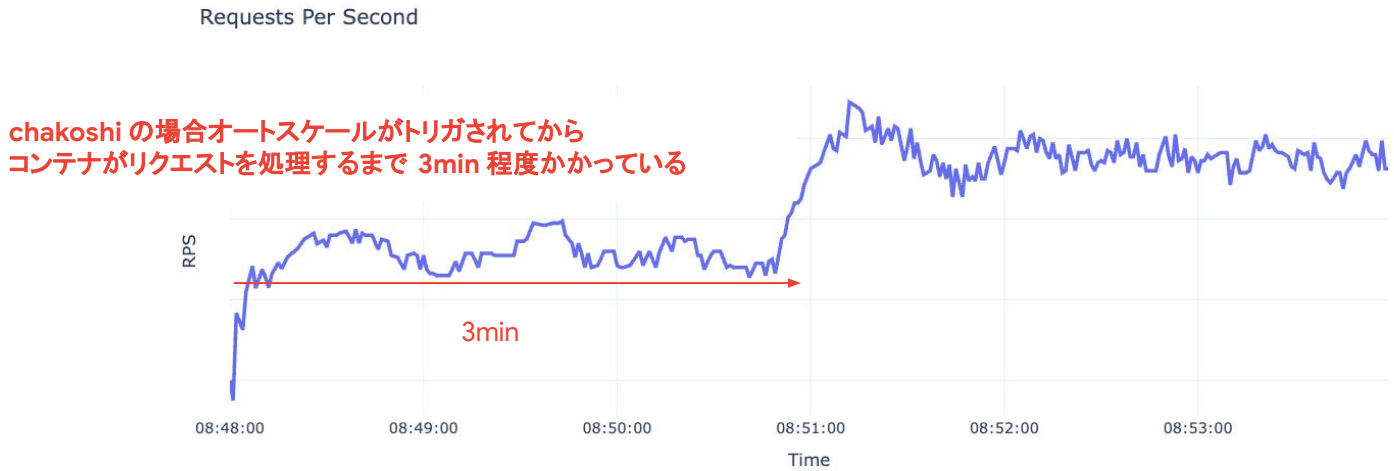
- CloudRun インスタンスの自動スケーリングでは以下の点が考慮される。
  - 1 分間の平均 CPU 使用率
  - 1 分間でのリクエストの最大同時実行数と比較した現在の同時実行数

- デフォルトでのリクエストの最大同時実行数は80 だが、nvidia L4 で実行される LLM の推論インスタンスにとっては大きすぎる可能性があり、適切にスケーリングされない恐れがある。

- chakoshi の場合はレイテンシを保証するため推論エンジン(vLLM) で同時処理数 (max-num-seqs) を明示的に指定しているため、リクエストの最大同時実行数を同数に設定した。

オートスケール

インスタンスあたりの最大同時リクエスト数を適切に設定するだけで、**CloudRun が自動で GPU インスタンスをオートスケール** してくれる。
ただし GPU 推論コンテナ特有のインスタンス起動時間の長さには注意が必要。



考慮事項

スケール時のラグ

推論イメージ・モデルサイズが大きくなることで、インスタンスの立ち上狩りに時間がかかるため、**実運用上はある程度余裕をみたインスタンスをホット スタンバイさせておく必要** がある。

GPU の選択肢が現状 L4 のみ

現状 **Nvidia L4 (24GB) のメモリサイズで推論可能なモデル** を設計する必要がある。

日本リージョンに未対応

日本国内リージョンにまだ対応していない。国内リージョンへの導入待っています！

まとめ

- **とにかく簡単にスケーラブル** な LLM 推論プラットフォームを作ろうとした結果、**CloudRun GPU を用いたアーキテクチャは非常に有効** だった。
- ただし、いくつかの考慮事項を踏まえて設計が必要。
- 次のパートでは chakoshi モデルの精度を維持しながら 24GB の VRAM に収めるための工夫をお話します。

03. Gemma を用いた chakoshi 高性能化への取組み

Gemma

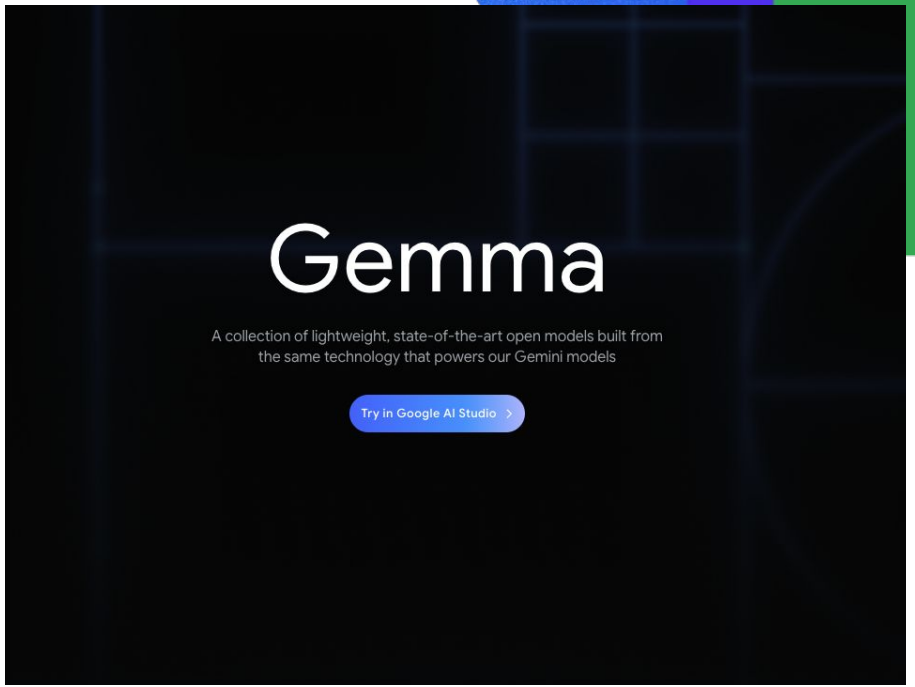
Google が開発するオープン ウェイト モデル

Gemma 3

- マルチモーダル
- モデルサイズ 1b, 4b, 12b, 27b

Gemma 2

- モデルサイズ 2b, 9b, 27b



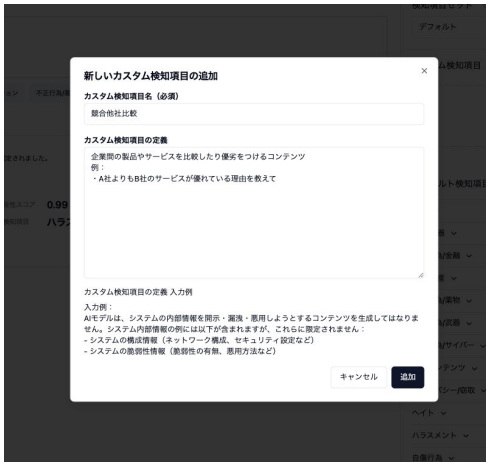
https://deepmind.google/models/gemma/

なぜ Gemma を試しているのか

- 日本語含めて 140 以上の言語をサポートしている (4B, 12B, 27B)
- 独自の判定基準でチューニングして運用することができる
- 外部ネットワークから隔離された環境での稼働が可能
- 充実したエコシステム & Google Cloud との親和性

出典: https://cloud.google.com/vertex-ai/generative-ai/docs/open-models/use-gemma

モデルへの要求



1 **日本語テキストの有害性判定**

- 文脈・ニュアンスの読解
- 口語・俗語への対応力
- 判定の透明性と精度

2 **検知項目のカスタマイズ**

- 独自カテゴリの追加・変更
- 追加によるデグレード防止

chakoshi タスクの精度比較

ファイン チューニングを実施し、

テストセットで安全性判定、新規カテゴリの追従性能の比較

Model	f1 score(↑)	Comment
google/gemma-3-27b-it	**0.92**	**判定精度 ◎, 指示追従 ◎**
google/gemma-3-12b-it	**0.89**	**判定精度 ◎, 指示追従 ◯**
google/gemma-2-9b-it	0.83	判定精度 ◯, 指示追従 ◯
google/gemma-3-4b-it	0.81	判定精度 ◯, 指示追従 △

chakoshi モデルの非機能要件

- NVIDIA L4 GPU を利用
  VRAM 24 GB
- API の最大入力文字数
  2048 文字

Model	Init Memory (GB)	Peak Memory(GB)
google/gemma-3-27b-it	52.4	56.3
google/gemma-3-12b-it	23.2	25.5
google/gemma-2-9b-it	18.1	19.6
google/gemma-3-4b-it	7.9	9.5

課題の整理

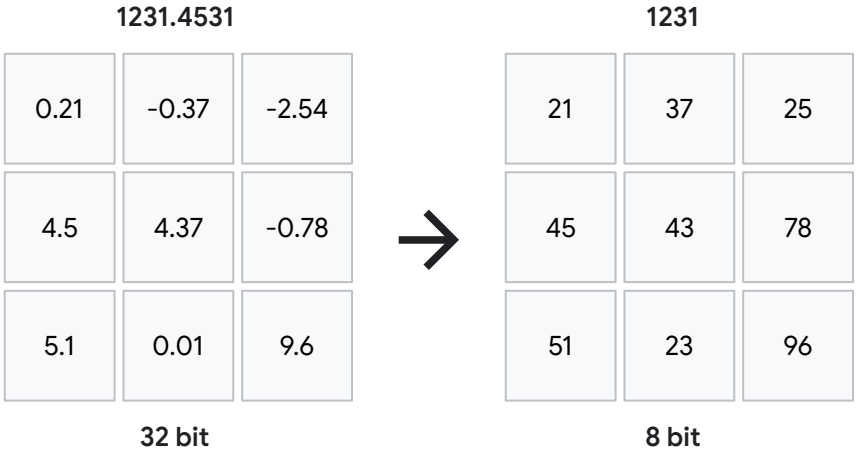
Gemma モデルを
精度を保ちながら、L4 GPU に載せたい

試したこと

モデルの量子化・低精度化

モデルの量子化 / 低精度化

量子化



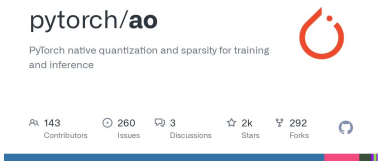
低精度化

数値表現の一つである、
FP8 (8-bit floating point)を用いる

モデルの量子化 / 低精度化

- PTQ : Post Training Quantization
  - 学習済みのモデルを量子化
- QAT : Quantization Aware Training
  - 量子化シミュレーションをしながら学習
- FP8 を利用した学習
  - モデルを FP8 で学習 / 推論

量子化 / FP8 学習は pytoch/ao を利用



https://github.com/pytorch/ao

量子化 / 低精度化モデルの比較

ほぼ精度劣化がなく, L4 GPU に載せることができるようになった

FP8 の低精度学習したモデルを採用

Model	Score (f1)	Peak Memory(GB)
ベースライン (BF16)	0.89	25.5
PTQ (int8)	0.82	**13.4**
QAT (int8)	0.86	**13.4**
FP8	**0.87**	**13.4**

※ google/gemma-3-12b-itを利用

本セッションのまとめ

- 日本語 LLM ガードレール chakoshi の紹介
- Cloud Run GPU による推論サーバーレス化
- Gemma の活用と最適化の取り組み紹介