Bidding and Auction Inference

Google Privacy Sandbox

Akshay Pundle

Apr 2024

Agenda

- Background
- Overall flow
- Technical details
- Q&A

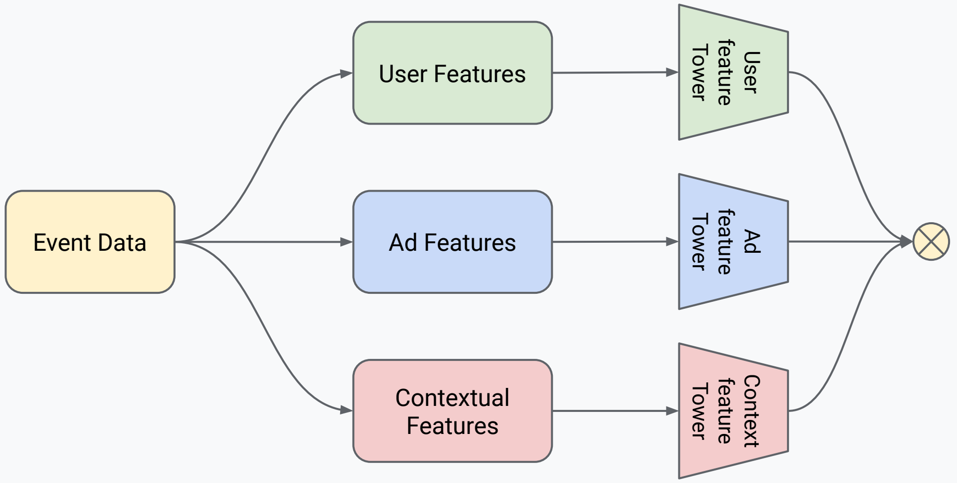
Background

Background

- Problem
  - Currently only models embedded inside JS / WASM are supported
  - Support first class capability to serve models inside TEE
- Adding Inference capability to Bidding and Auction services inside TEE
  - Initial support is for Protected App Signals on Android (Beta - EOQ2 2024)
  - Support for Protected Audience (Android, Chrome) to be added as a follow up (ETA TBD)
  - Initially supported on Bidding servers, can expand to others TEE servers in the future
- Scope for this presentation
  - Limited to serving models which are already trained
  - Does not cover model training
- Status
  - This is work in progress, seeking feedback from the larger community
  - Core pieces of implementation are available, but are not ready for production traffic

Background

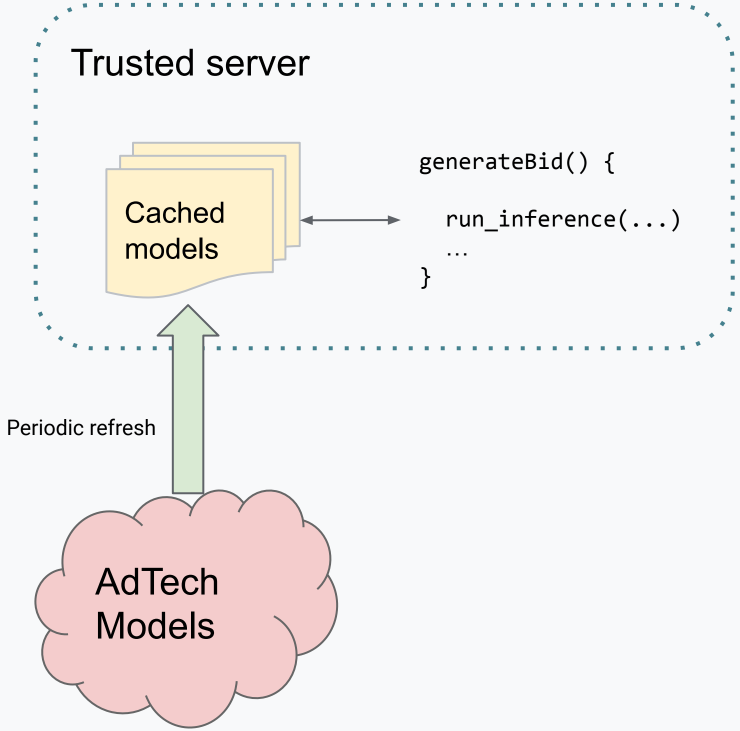
- Inference capability includes
  - Load models from cloud bucket
  - Call inference from JS worklets (e.g. `generateBid`, `prepareDataForAdRetrieval`)
  - Execute inference / predictions with standard backends (Tensorflow, PyTorch) inside TEE
- Support for factorized models
- For details, see Inference explainer



Overall flow

Executing models in B&A

- Core functionality is to load and execute models
- This will support factorized and non-factorized models
- Models will be callable from JS
- Models will be periodically refreshed

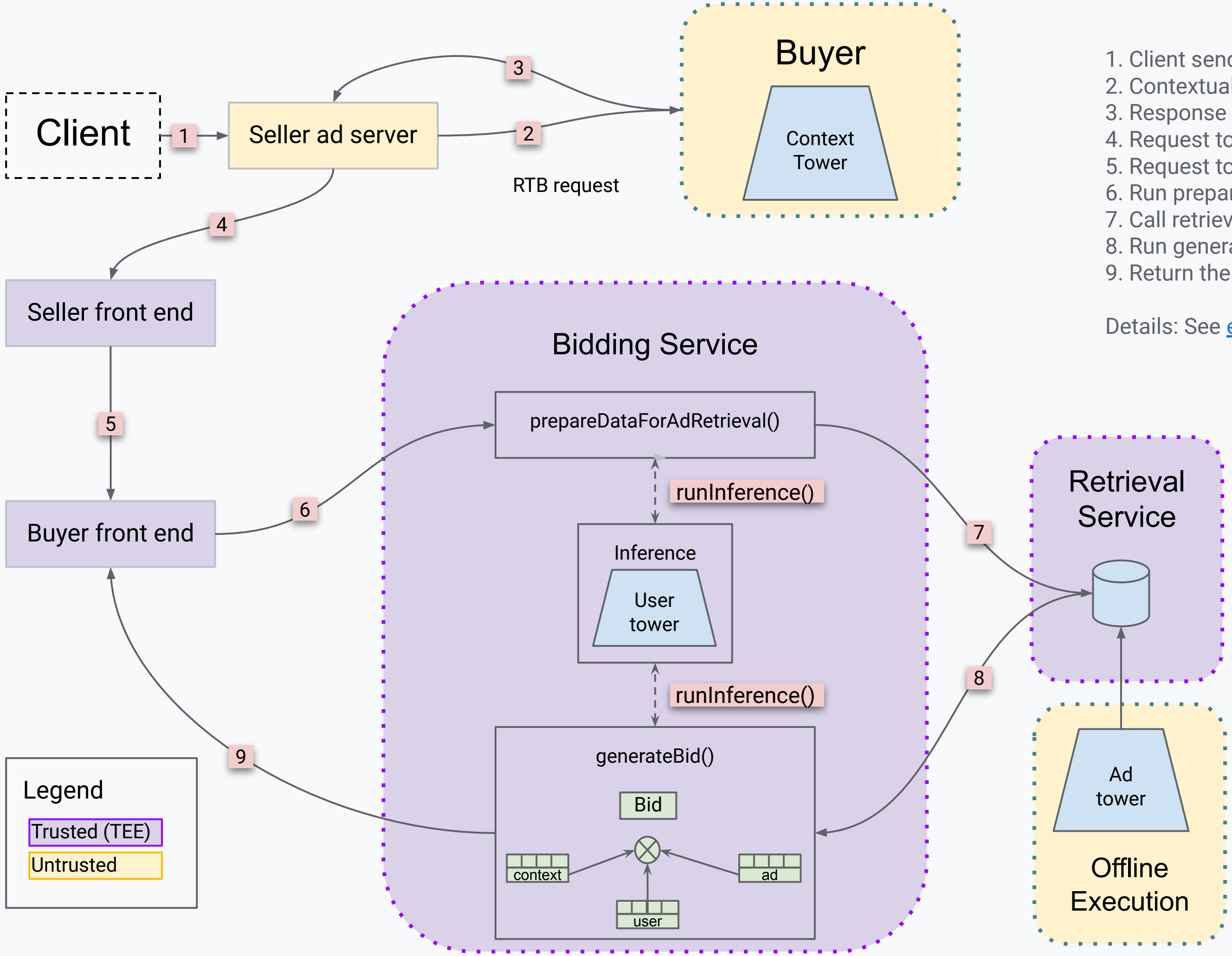


```
generateBid() {

  run_inference(...)
  ...
}
```

Factorized model flow

- B&A will support different plugin points for factorized models
    - B&A servers can do predictions with ML models inside TEE
    - RTB requests as a part of Contextual request (outside TEE) can generate contextual embeddings and pass to the `generateBid` function.
    - Pre-generated embeddings can be fetched from the retrieval server and be passed to `generateBid`. In this case, the model runs offline outside the TEE. The generated embeddings are uploaded to the retrieval server.
- Data flows for transferring embeddings to `generateBid` and consistent versioning of models and embeddings will be supported
- Ad-tech can use this support as building blocks to create flows for their inference



1. Client sends request
2. Contextual request to buyer
3. Response with buyer_signales
4. Request to SFE
5. Request to each buyer's BFE
6. Run prepareDataForAdRetrieval()
7. Call retrieval server to get ads
8. Run generateBid() for all ads
9. Return the calculated bid

Details: See explainer.

Versioning

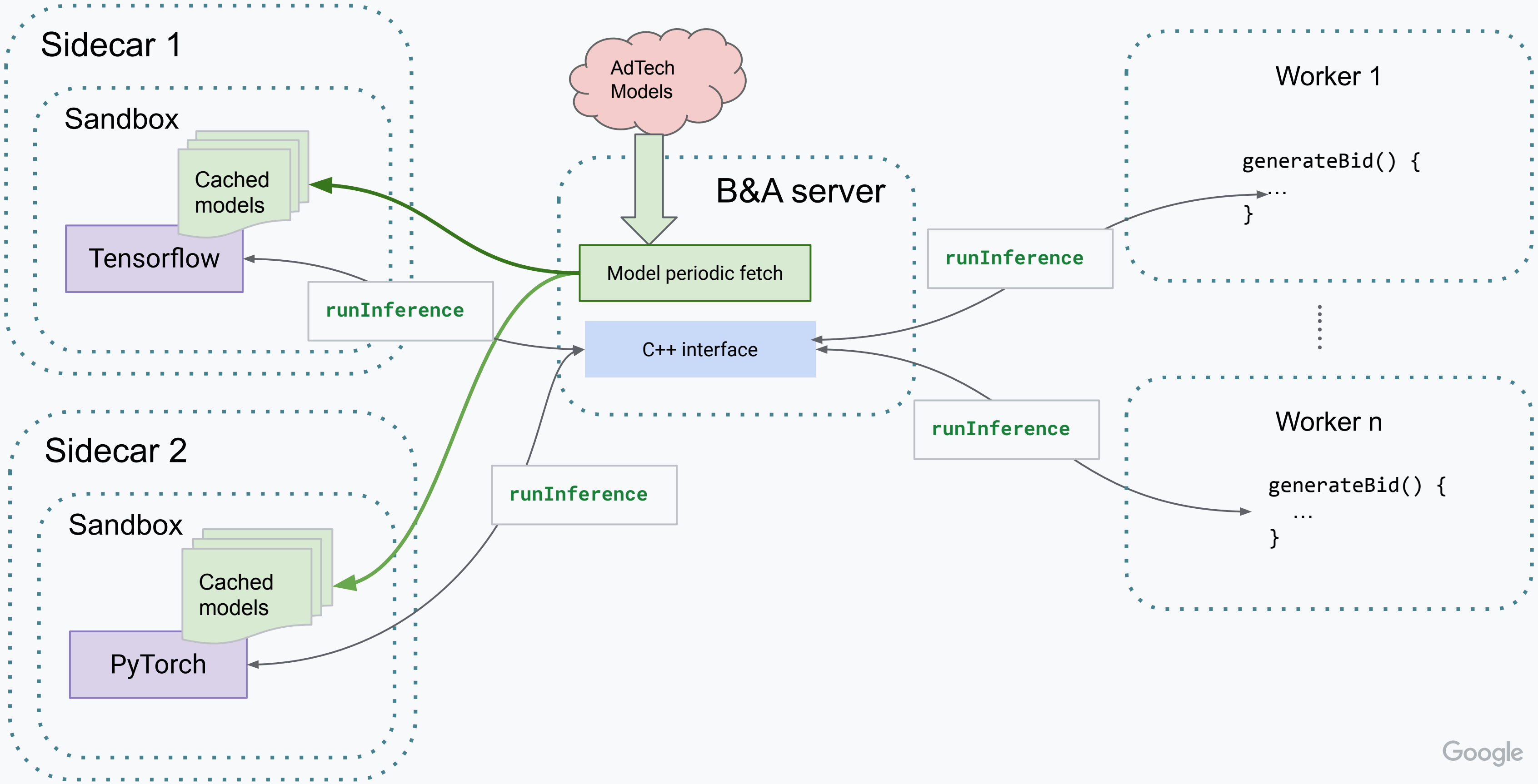
- Important to keep the versions of embedding and models in sync
- Passing version information supported between contextual path, retrieval server and the UDFs (e.g. `generateBid` and `prepareDataForAdRetrieval`)
- Specific versions of embeddings can be coordinated through this version information.
- Model versions can be encoded in the path.
    - Path is the main primitive for identifying a model identifier
    - The path can encode the name and version information together
    - Paths of all loaded models are made available to the UDFs, so the right version of the model can be selected
- Details: Versioning section in the Inference explainer

Technical details

Model execution

- Two main ways of executing models inside TEE
    - Embedded models
    - Externally provided Models for supported backends (initially Tensorflow and PyTorch)
- Embedded models are fully contained inside the ad tech JS / WASM
- Non-embedded models for supported platforms will be executed by the B&A code by linking C++ libraries for each model type
- Initially Tensorflow and PyTorch (Libtorch) are supported. Support for other systems (e.g. ONNX) can be added later based on ad tech demand.
- Initially models will run on CPU. As accelerators become available we will explore integration - timelines are TBD

Sidecar model execution



API

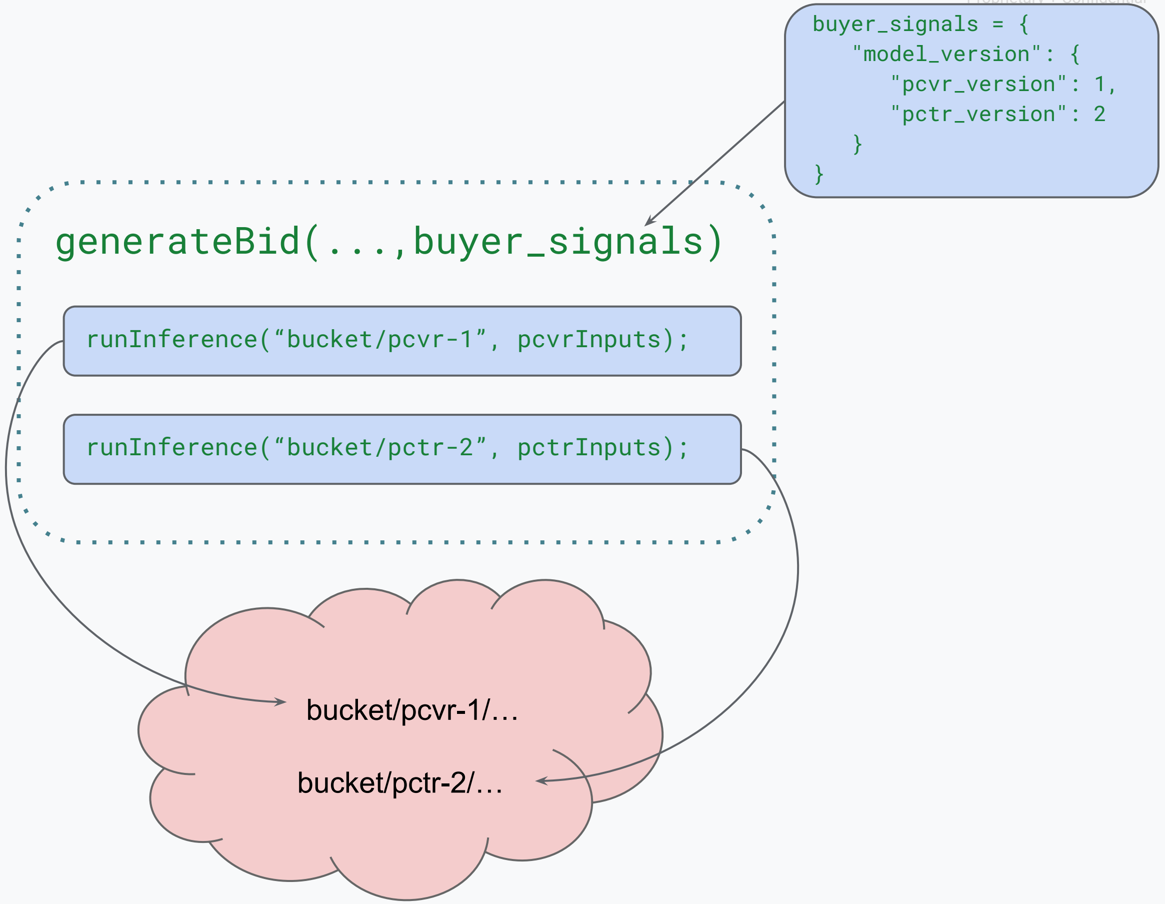
- `runInference(request)` is the main way of calling inference.
- Batch API supports calling multiple models in the same request
- For each model, inputs and outputs can be arbitrary tensors.

Request for one model:

```
"model_path": "my_bucket/models/pcvr/1/",
 "tensors": [{
     "tensor_name": "user_signal_1",
     "data_type": "DOUBLE",
     "tensor_shape": [2,1],
      "tensor_content": ["0.454920","-0.25752"]
  }]
```

Model loading

- Models will be loaded from cloud buckets
- Once models are loaded, they can be used for inference
- The main way to address a model is by its path
- Versioning info can be used inside `generateBid` (and other UDFs) to create the correct model path.
- There is an API that can be used from `generateBid` to get a list of loaded model paths.



```
buyer_signals = {
   "model_version": {
      "pcvr_version": 1,
      "pctr_version": 2
   }
}
```

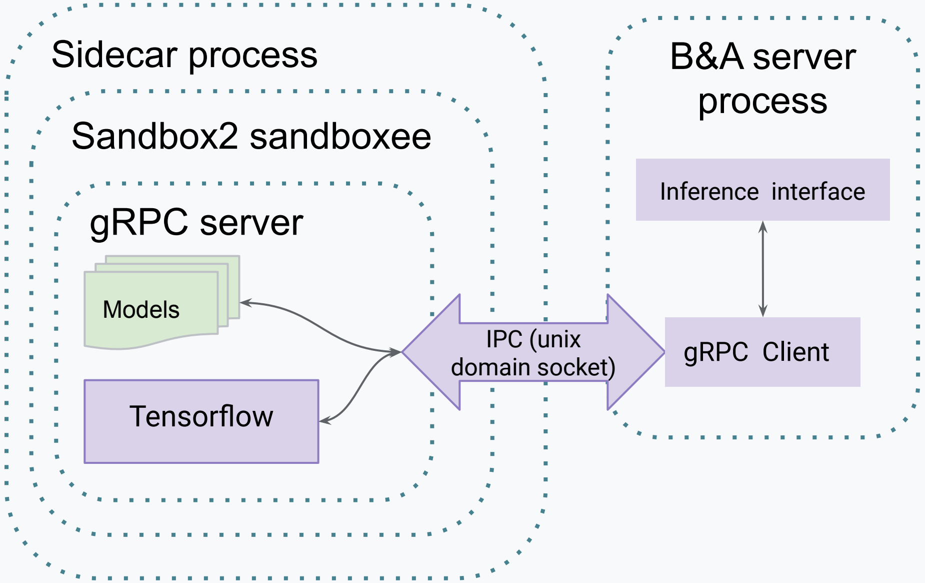
```
generateBid(...,buyer_signals)

runInference("bucket/pcvr-1", pcvrInputs);

runInference("bucket/pctr-2", pctrInputs);
```

Sandboxing

- Inference runs in a separate process called the inference sidecar
- The sidecar is a separate binary compiled with one backend execution system (e.g. a specific version of Tensorflow).
- The ML backend is sandboxed using Sandbox2.
- Model loading and inference happens only inside the sandbox
- Inside the sandbox, we run a gRPC server that receives requests and responds to the host binary (e.g. Bidding server).
- gRPC Communication between the host and the sidecar happen through a unix domain socket managed by Sandbox2.



Privacy mitigations

- Privacy mitigation are still in progress and will be put in place before MVP
- Models and ML backends are completely sandboxed
- Models will periodically be reset to their original state to make sure they are not stateful
- More details will be added to the explainer in the future

Q&A

Feedback themes

- Preferred ML backends
- Model size, scale, performance
- Model versions and update frequency
- Operational and quality metrics
- Cost and Utility
- Detailed questionnaire: [this doc]
- Feedback: https://github.com/WICG/protected-auction-services-discussion/issues/59

05

Appendix

Google

References

- Inference explainer
- Feedback questionnaire
- Bidding and Auction services explainer
- Protected app signals explainer
- Protected Auctions explainer