Hybrid AI for the Web: Caching

Michael McCool, Geoff Gustafson,
Sudeep Divakaran, Muthaiah Venkatachalam

Intel

13 June 2024, W3C WebML WG

Outline

- Key points from offline discussion…
- Model size and download times
- Working set size and size of cache needed
- Security and privacy considerations
- Caching desired properties
- Possible solutions
  - ***No silver bullet!***
  - Some options, but with tradeoffs

Key Points from Offline Discussion…

- Some models are too large to download during session
  - Need something like Background Fetch
- Rather than specific models, perhaps think about particular use cases
  - Functionality could be implemented with one of several models
  - Somewhat equivalent to “fixed-function APIs”
- Adapters and variants are a challenge
  - Many models have them “baked-in”
  - Models may have many variants that differ in quantization, etc.

Security and Privacy Considerations

- ***Current browsers implement only per-origin local caches***
- Cross-site privacy risk based on cache timing analysis:
  - Site A can figure out if a user visited Site B
- Per-origin caches tolerable for “typical” (non-AI) web resources:
  - Sharing rate for images is low in practice
  - Files that are often shared tend to be small, e.g. script libraries
- **BUT AI Models are large *and* potentially shared**

- Arbitrary key-value cross-site caches are also a privacy risk
  - Data exfiltration and tracking

Possible Mitigations

1. Disallow use of WebNN in third-party context by default
   - *Already part of WebNN specification*
2. Generate keys (e.g. use a hash) based on actual model content
   - Avoids data exfiltration (block data transfers)
   - … but possibly not tracking (needs only existence checks)
3. Limit number of built models and/or cache checks
   - Avoid use of multiple model existence checks for transferring many bits

Caching Desired Properties

1. **Reduce Latency:** Fetch model from cache upon second use
2. **Reduce Bandwidth:** Avoid redundant downloads
3. **Reduce Storage:** Consolidate and reuse models as much as possible
   - Cross-site?
   - Across implementations?
   - Model consolidation?
     - Across equivalence classes? (e.g. different quantization levels of same model)
     - Across different serializations?
4. **Preserve Privacy**

Proposal: Define New Model-Aware Caches

Some key ideas:

1. Use “**fake misses**” (delays) to avoid redundant downloads.
2. Progress model loads/timers only when requesting page is **inactive.**
3. Identify cache items by content-dependent **hashes.**
4. Use **deduplication** to avoid redundant storage.

Some alternatives:

1. Use existing APIs/caches, perhaps with some extensions
2. Use the File System API + Background Fetch

Prototype Status

- Implemented:
  - Node cache with hashes as keys, external Redis service for storage
  - **However:** Model cache seems to be more generally useful
- Next Steps:
  - Implement *model* cache
  - Base on Service Worker Cache, Background Fetch if possible
  - Three implementation options:
    1. Capture/replay graph building by wrapping WebNN API (shim+extension)
    2. Modify the implementation, e.g. Chromium, “under the hood” (best for performance)
    3. Cache an existing model serialization, and use a model loader
  - Write a detailed proposal document and explainer…

Backup

AI Model Download

Average Home Network Speeds

- 90 Mbps – Global
- 216 Mbps – US

Sources:

- Speedtest.net
- USA Today: What is a Good Internet Speed

Model Size vs. Download Time

Maximum download in 1 minute:

- Global: 675 MB
- US: 1642 MB

Time to download Phi-3-mini:

- 3.8B bfloat16 parameters
- 2*3.8B = 7.6 GB
- Global: 11.3 minutes
- US: 4.69 minutes

How Many Models need to be Cached?

- Number of models on Hugging Face (as of 2024-05-28):
  - 628,216
- *Most* of these are not directly useful for web applications
  - Research projects
  - Not well-tuned or aligned
  - Components of other models
  - Inappropriate use cases

Assumption (?):

- There are 250 “useful” models.
- Storage required for 250 models the same size as Phi-3-mini:
  - 1.9TB

BUT:

- Many variants of each model
  - Quantization, encoding, number of parameters
- Many derivatives of each model
  - Fine-tuned
  - Adapters, if used, may not be separate
- We don’t know the working set size (a dozen models?)
- Storage for 12 models the same size as Phi-3-mini:
  - 91.2GB

Security and Privacy Considerations

- **Bad:** Arbitrary key-value pairs in a shared cross-site model cache
  - Can be used for “mega-cookies” to exfiltrate data!
  - Can be also be used as trackers.
- An abuser could build a fake model
  - Embed data to be shared in the model
- Then the attacker would store the fake model in the cache.
  - Attacker can retrieve model based on key from a different site;
  - Then probe model to recover data.

NOTE: ***Service Worker Cache API cannot be simply made cross-origin.***

References

- Storage Partitioning (see HTTP Caches especially)
- GPU Web Privacy Considerations (shader caches)
- Felten and Schneider, Timing Attacks on Web Privacy, 2000
- Judis, Say goodbye to resource-caching across sites and domains, 2020
- CloudFlare (CDN) Origin Cache Control (can also be enabled in CDNs)
- Background Fetch – related API for large downloads.
- Cache AI models in the browser (Google) – how to use existing per-origin cache mechanisms for AI models