June 2025

Optimizing Memory Consumption in Chrome

Muhammad Qasim Atiq Ullah | Faizan Sohail

Agenda

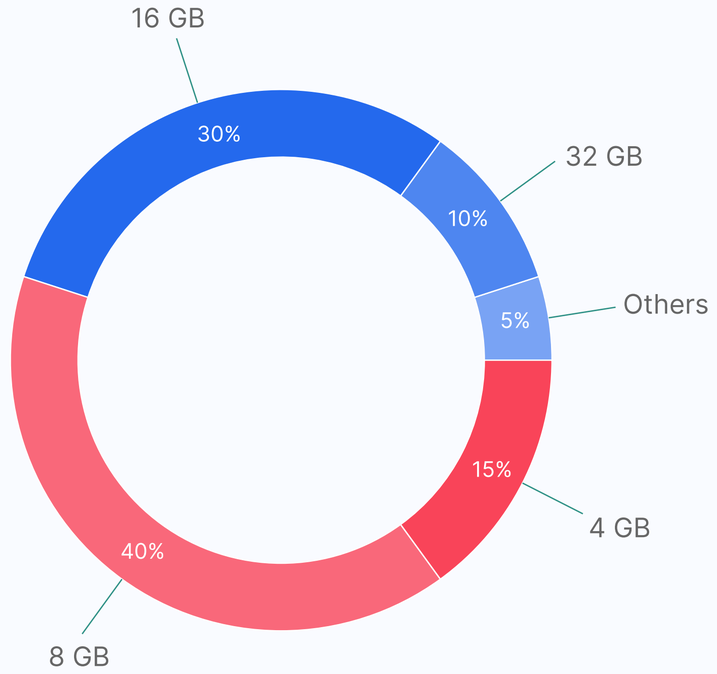
1. Problem
2. Previous Solutions (Memory Saver)
3. Our Solution (Dynamic Discarder)
4. Results
5. Discussion & Conclusion

Growing Demands

With rising demand for multi-tasking and resource intensive tasks - systems should also be sufficent enough to cater these needs.

How much ram is enough?

How much is browsing prevalent?



Global Market Segments of RAM in Laptops

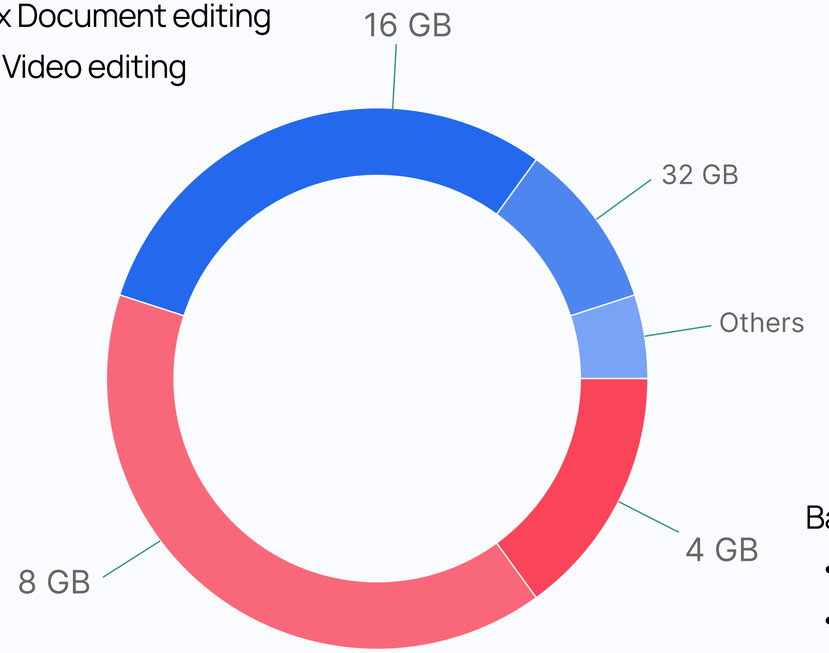
Browsing

Intermediate Needs:

- Heavy Multi-tasking
- Moderate Gaming
- Complex Document editing
- Photo & Video editing

How much RAM is enough?

It depends...



Global Market Segments of RAM in Laptops

Basic-Intermediate Needs:

- Moderate Multi-tasking
- Web Browsing
- Document Editing
- Light-photo editing

Basic Computing Needs:

- Light multi-tasking
- Web Browsing
- Document Creation

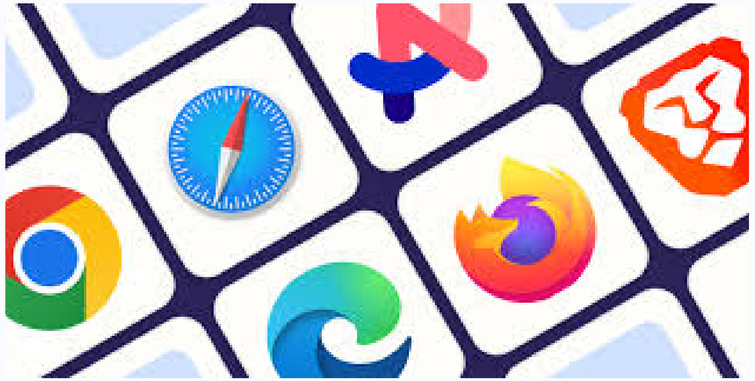
Sources: HP, Lenovo, PC World

Browsing

Chrome

With the rise in remote work and digitalisation browser usage is like it was never before.

Palo Alto's research on Browser Usage across companies showed 85% of the workforce's day was on browser.



Survey by ESG in 2025 concluded 94% of knowledge workers spend at least half of their productive time inside a browser

Chrome

Usage

Chrome accounts for Majority 65.7% of the Browser Market Share

While convenient in usage it consume a lot of RAM owing to its multi-process design.

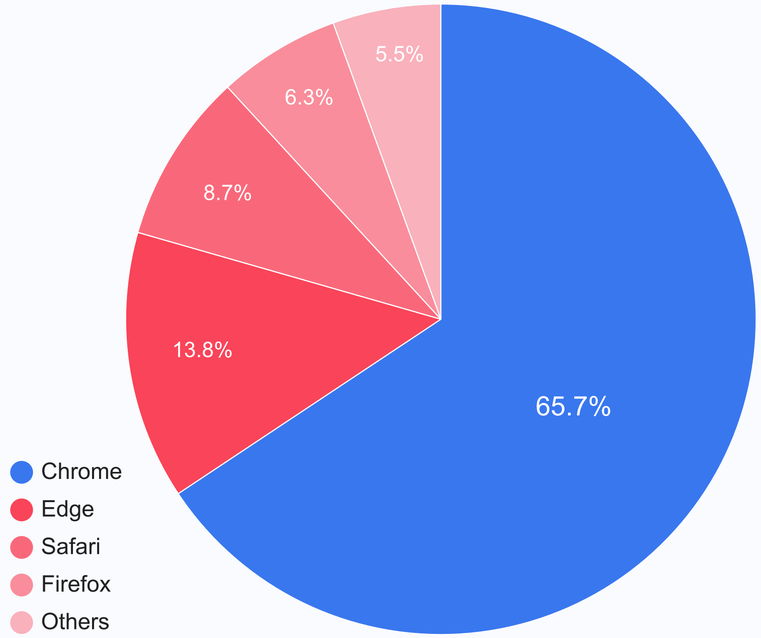
Cloudzy ranks Chrome as one of the most RAM consuming browser

Consumes approx. 1 GB RAM for 10 tabs and scales to 1.9 GB for 20 Tabs

Why?

To ensure smooth and fast performance

But does it take system constraints into account?

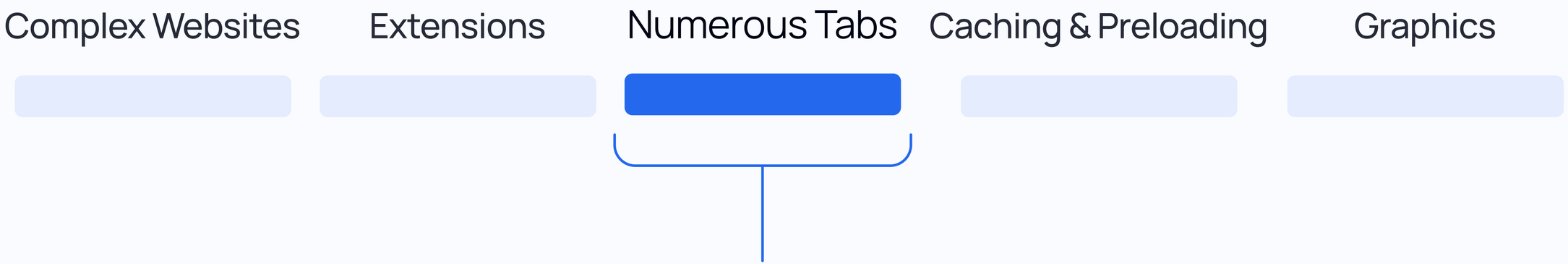


Source: Backlinko

> I’ve noticed that Google Chrome has been consuming a huge amount of memory, even with just a few tabs open. Has anyone else experienced this recently? Are there specific extensions or settings that might be causing the issue? Any tips to optimize Chrome’s performance would be appreciated

- David 4368 (Google Chrome Help) May 15,2025

Different factors consuming Memory



Add-on interactive features, videos, fancy pictures.

Working in background. Heavy workloads - updating webpages, AdBlockers.

Each tab operates with independent processes, multiple processes stack up memory

Stores parts of Websites and pre-loads website expected to be visited next.

Memory is used at times to display things smoothly.

Solutions

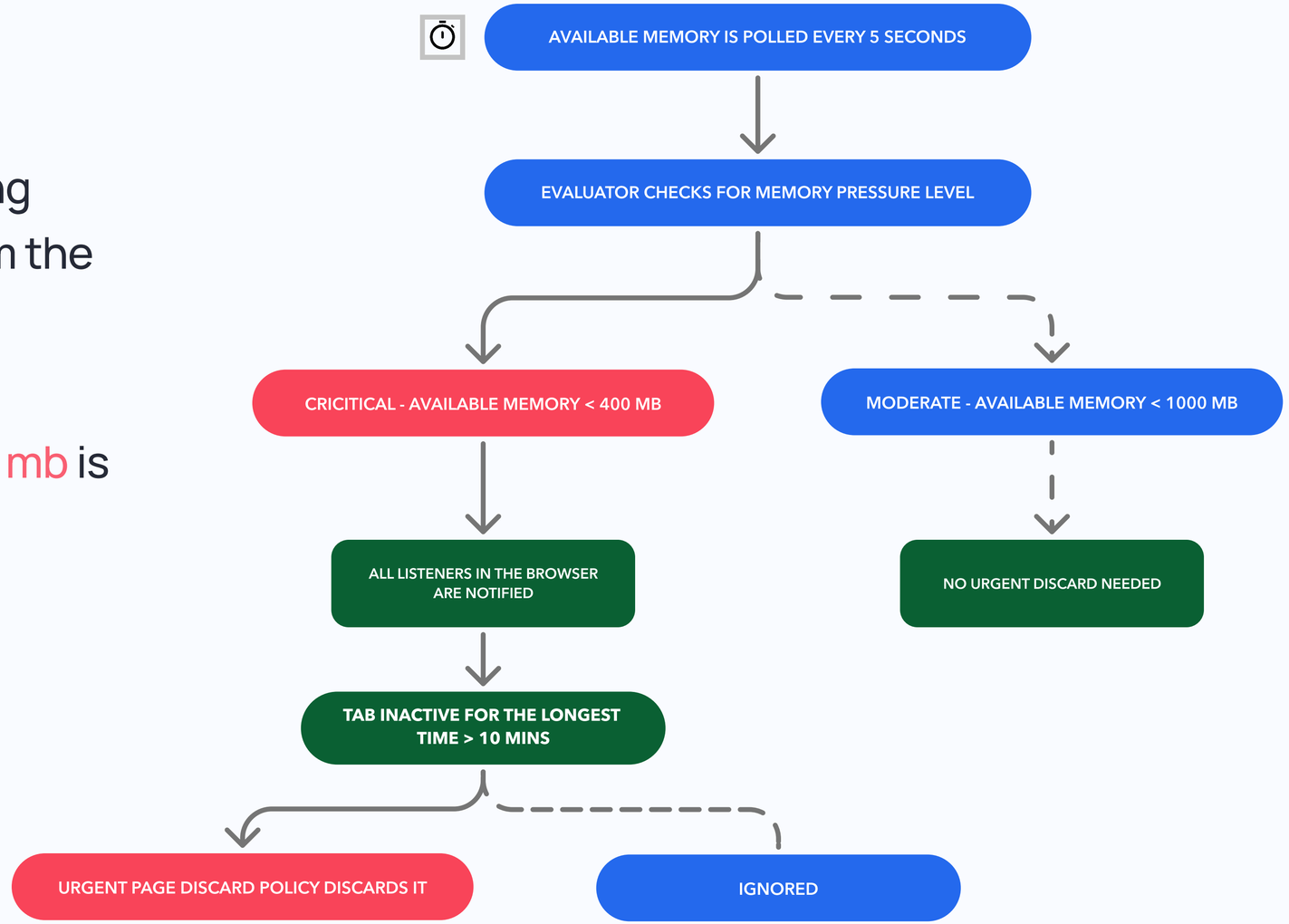
1. Load Based Discarding
2. Memory Saver
3. Dynamic Discarder

Load Based Discarding

Introduced in 2015, load based discarding continuously discards tabs starting from the longest inactive time.

Critical Pressure hit when less than 400 mb is available.

Reaches deadlock if all tabs are recently accessed.



Memory Saver

Discards Inactive Tabs

Removes the content of the tab from memory and upon re-visiting the tab the contents are reloaded.

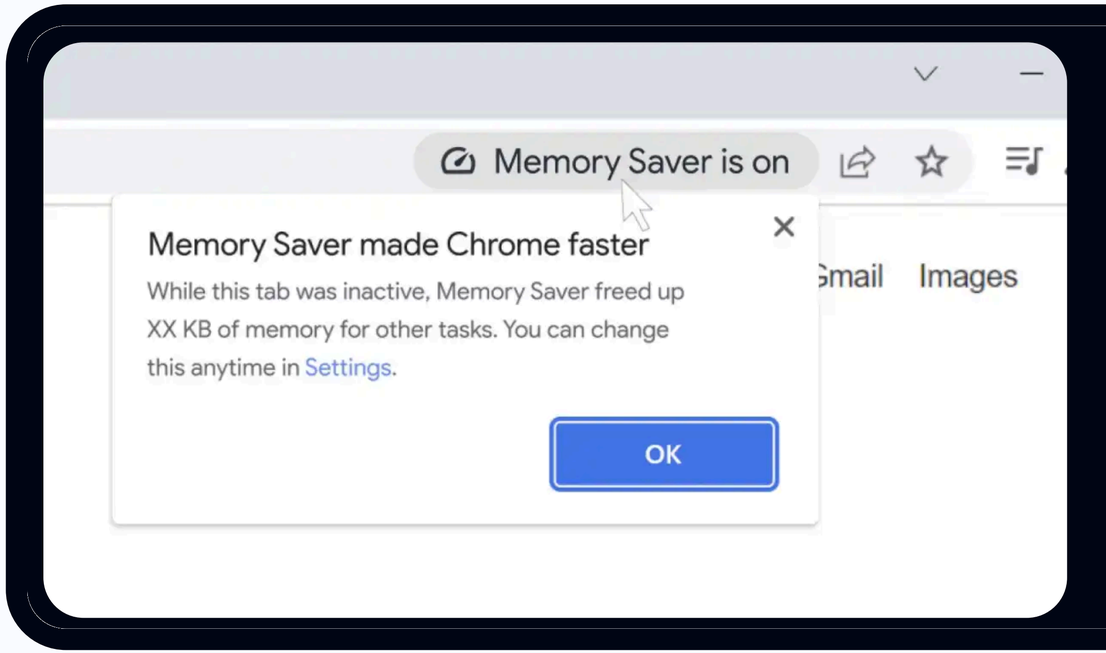
Reactive + Proactive

Building upon its previous implementation to discard tabs under critical memory levels, tabs are discarded as per user behaviour.

Unused tabs no longer stored indefinately

Tabs unvisited for long intervals are selected for Discard

Uses up to 40% less memory



Sources: Google

How did it come to be?

1

High Efficiency Mode

Foundation

Introduced in Dec, 2022 this feature would discard tabs automatically after an interval of 2 hours of tab inactivity.

2

Heuristic Mode

Experimental Feature

Motivation: Discarded tabs revisited within ~24 hours about ~60% of the time.

The Probabilistic Memory Feature aimed on improving this by better selecting which tabs to discard.

Initiated in August, 2023 - Abandoned in Jan, 2024

3

Multi-Mode

Current Implementation

Built upon the results of the PMSF this mode was introduced in November, 2024.

It discarded tabs across three levels varying in behaviour for Moderate, Balanced and Maximum memory savings.

Source: Chromium Issues & Google Blog

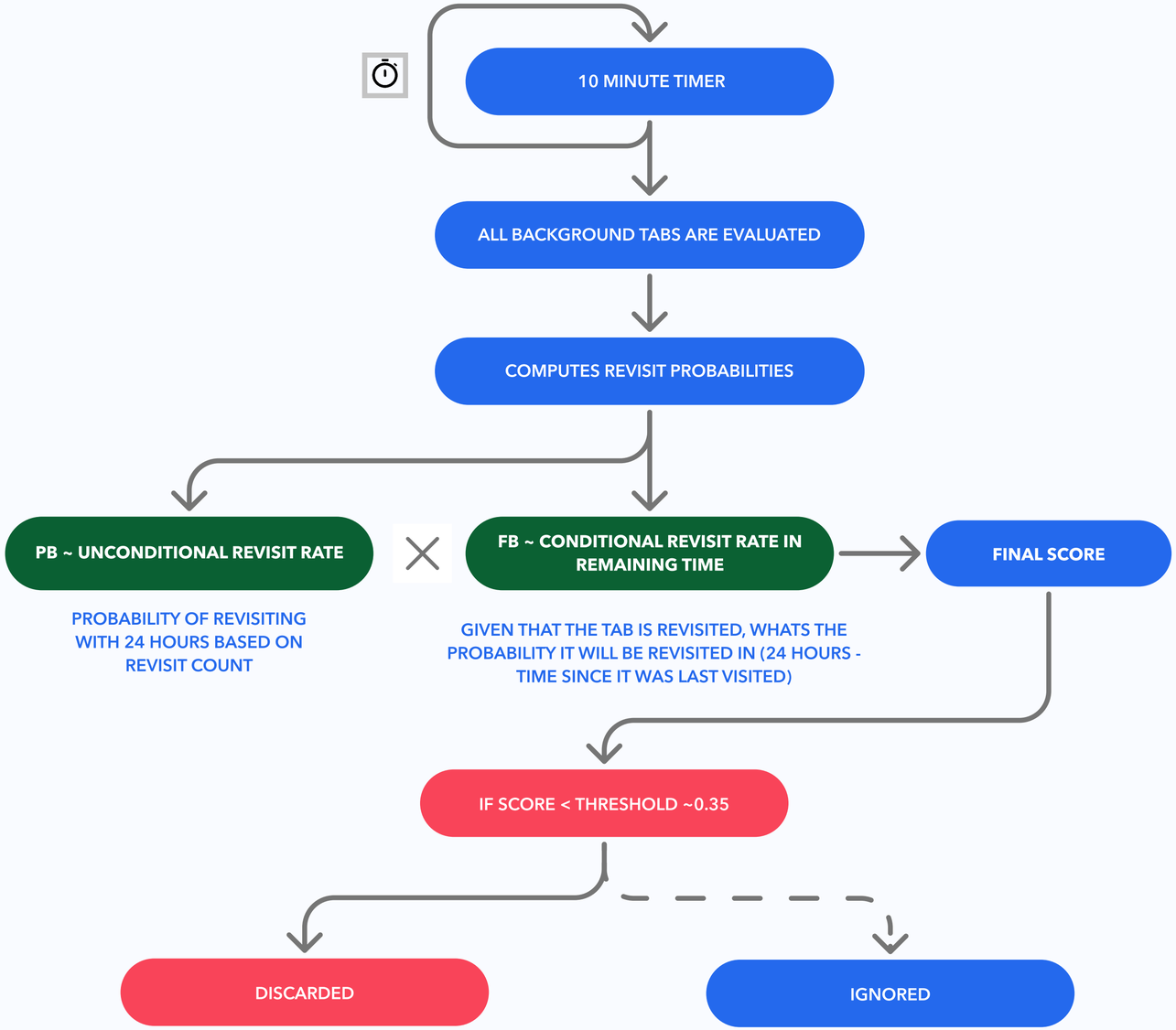
PMSF

Made decisions based on Predicted tab reuse likelihood scores.

Given tab states this mode used Probability Distributions of revisiting a tab using UMA metrics

Similar CDF containers based on number of revisits

Files cleared in Early Jan 2024

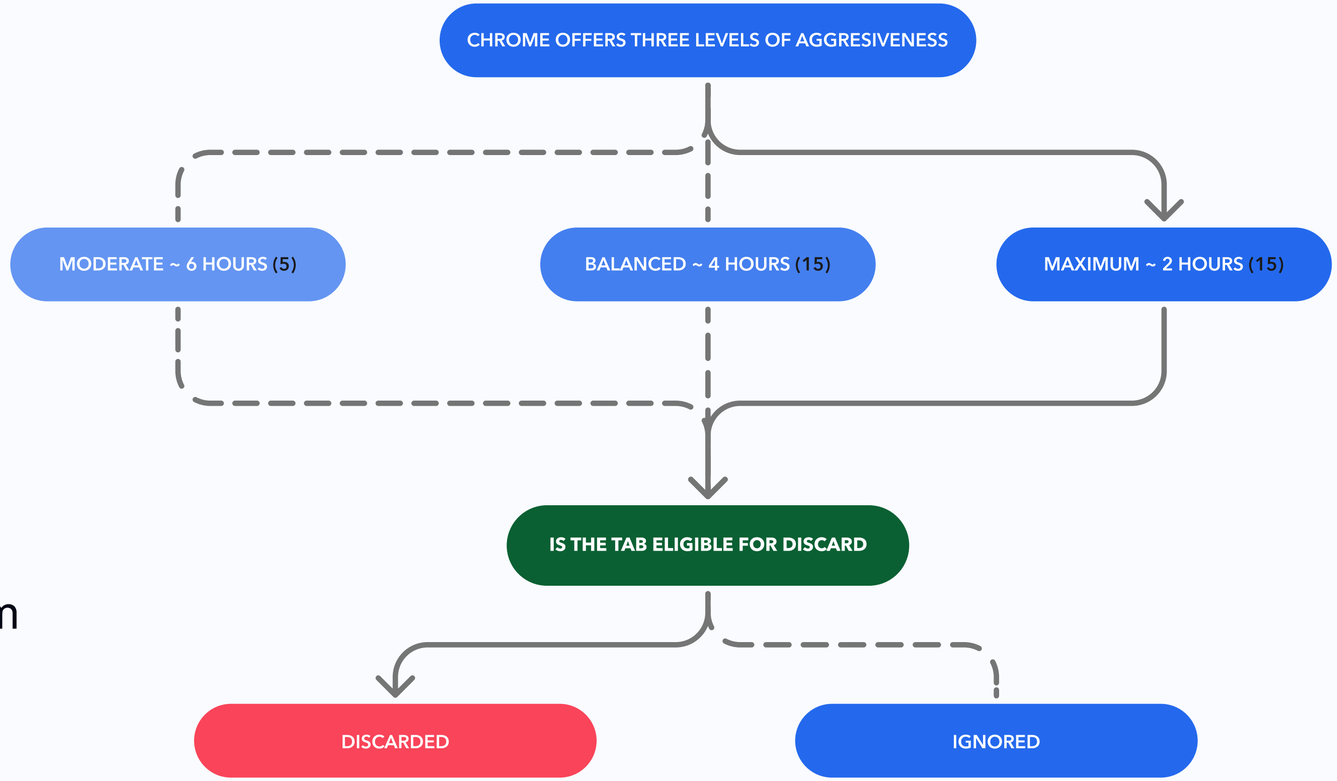


Multi-Mode

Built on top of the PMSF, allowing for explicit control over discarding behaviour

Tabs are discarded after the following amount of time passes in activity

Tabs frequently accessed are exempted from the discarding mechanism - based on number of revisits.



Here are the Eligibility Checks

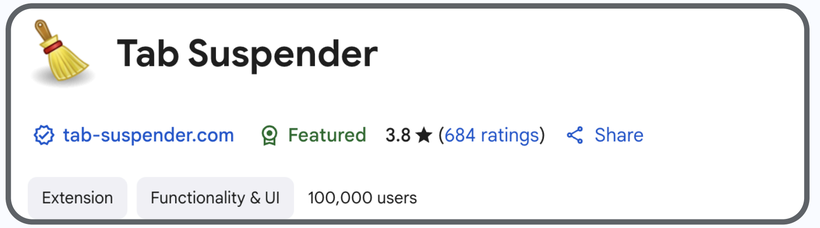
- Tab exempted
- Tab is visible
- Tab is playing Audio
- Recently Audible/Visible
- Tab is hosting a pdf
- Notifications Enabled
- Extension Protected
- Capturing Video/Audio
- Connected to External Device
- Updating title/favicon
- Form Interactions
- User edited tab content
- Using Devtools

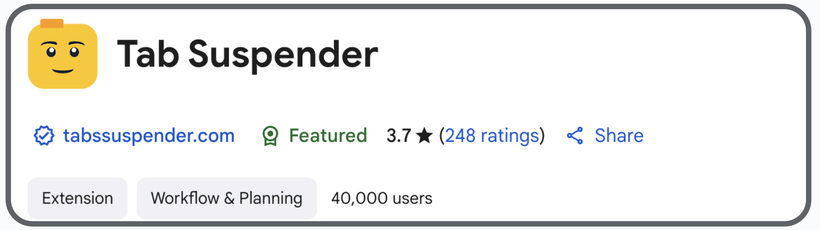
Extension Based Discarding

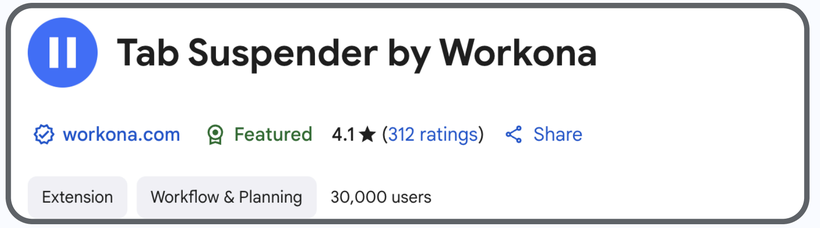
Chrome allows Extensions to discard tabs

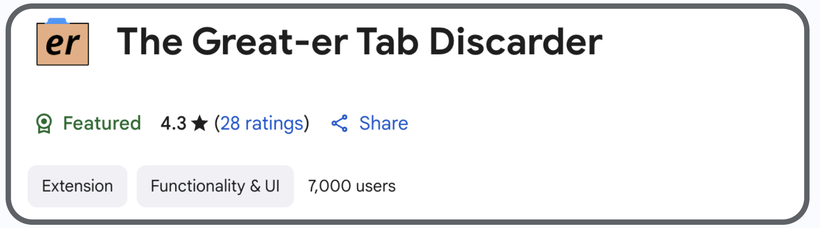
Extensions are able to bypass the Eligibility Criteria

Multiple extensions operate based on customised timing mechanism or manual attempt:









Towards an Intelligent Discarder

ML based Discarder

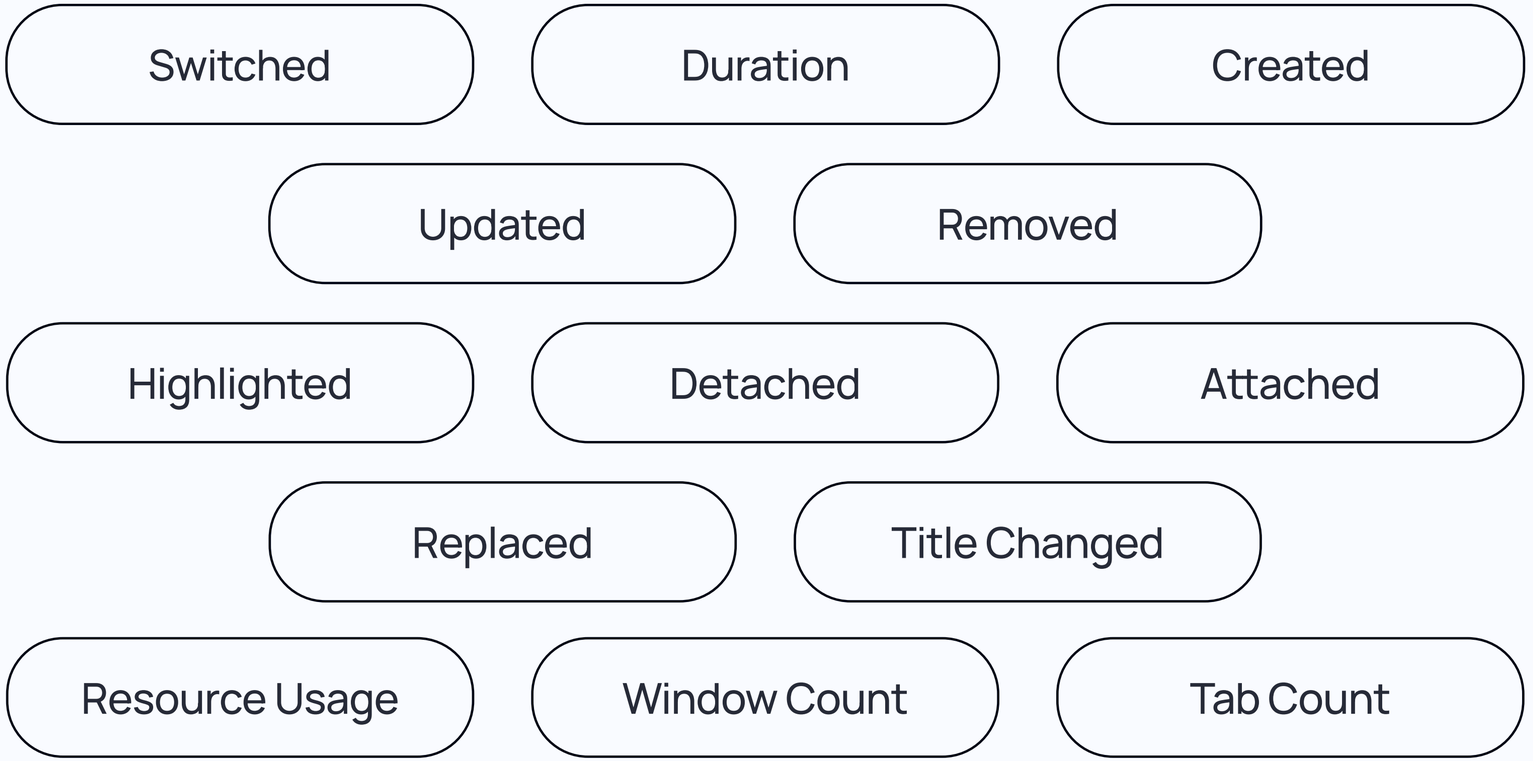
Given previous interactions with the Tab, the model predicts if it might be removed.

Discards based on this prediction.

Instead of focusing on future inactivity time and discarding for the time being - this approach minimises aggressiveness

Data Collection

The Following life-cycle events are recorded through the Chrome Tab API



Activity Logger

Collects Real-time event hooks, process snapshots allowing for comprehensive visibility into tab behaviour

Key Features: idle time.

Sample of records after Data Cleaning

	timestamp	type	data
16	2025-03-10 23:22:26	tabSwitched	{'type': 'tabSwitched', 'fromTab': None, 'toTa...
17	2025-03-10 23:22:26	tabHighlighted	{'type': 'tabHighlighted', 'windowId': 8379253...
19	2025-03-10 23:22:27	tabSwitched	{'type': 'tabSwitched', 'fromTab': 837925578, ...
20	2025-03-10 23:22:27	tabHighlighted	{'type': 'tabHighlighted', 'windowId': 8379253...
33	2025-03-10 23:22:57	tabUpdated	{'type': 'tabUpdated', 'tabId': 837925613, 'ur...

Every captured event is sent as a JSON payload to an external server endpoint over HTTP POST for centralised storage and analysis.

Setting up Models

Long Term Short Term (LSTM)

Computationally Efficient, Captures Temporal Dependencies

Supporting Thesis, Predicting Navigational Patterns
in Web Applications
using Machine Learning Techniques.

Trained LSTM, Transformer and CNN-LSTM
For Long Sequences LSTM generally outperforms the others. ~75% Accuracy at best.

Given a sequence of previous Tab Lifecycle Events (~15) the model trains to predict the next event - anticipating user behaviour.

Predict “Tab Removed” and thus discard that tab.

Replaces the coarse rules and tailors discards to real time behaviour.

Three Models

1

Vanilla - LSTM

Interprets tab activity purely on the order of events.

```
# Training Sequence Length = 10
# Model Architecture: Embedding(8) → LSTM(16) → Dense(vocab_size, activation='softmax')
# Hyperparameters: embed_dim=8, lstm_units=16, seq_length=10;
# Training Setup: optimizer=Adam(learning_rate=0.001), batch_size=32, epochs=50, validation_split=0.2, loss='categorical_crossentropy'
# Performance Monitoring: achieved val_acc≈63.26%, val_loss≈1.05;
```

2

Time - LSTM

Time Interval between events → Additional input feature.

```
# Training Sequence Length = 15
# Model Architecture: Embedding(16) → Time-Aware LSTM (32 units processing event+delta inputs) → Dense(vocab_size, activation='softmax')
# Hyperparameters: embed_dim=16, lstm_units=32, learning_rate=0.001, batch_size=32, seq_length=15
# Training Setup: optimizer=Adam(learning_rate=0.001), epochs=20, validation_split=0.2, loss='sparse_categorical_crossentropy'
# Performance Monitoring: achieved val_loss=1.03, val_acc: 65.3%
```

3

Attention - LSTM

Attention Mechanism over the LSTM output.

```
# Training Sequence Length = 15
# Model Architecture: Embedding(16) + Δt concat → LSTM(32, return_sequences=True) → Bahdanau-style attention (Dense(1) → Softmax) → context vector (weighted sum) → Dense(vocab_size, activation='softmax')
# Hyperparameters: embed_dim=16, lstm_units=32, learning_rate=0.001, batch_size=32, seq_length=15
# Training Setup: optimizer=Adam(learning_rate=0.001), epochs=20, validation_split=0.2, loss='sparse_categorical_crossentropy'
# Performance Monitoring: best validation accuracy ≈61.14% , val_loss=1.1775
```

Implementation via Extension

Local Flask Server

Runs on the user's machine, receives sequence data + predicts next event → communicates with the Extension.

When to initiate the process?

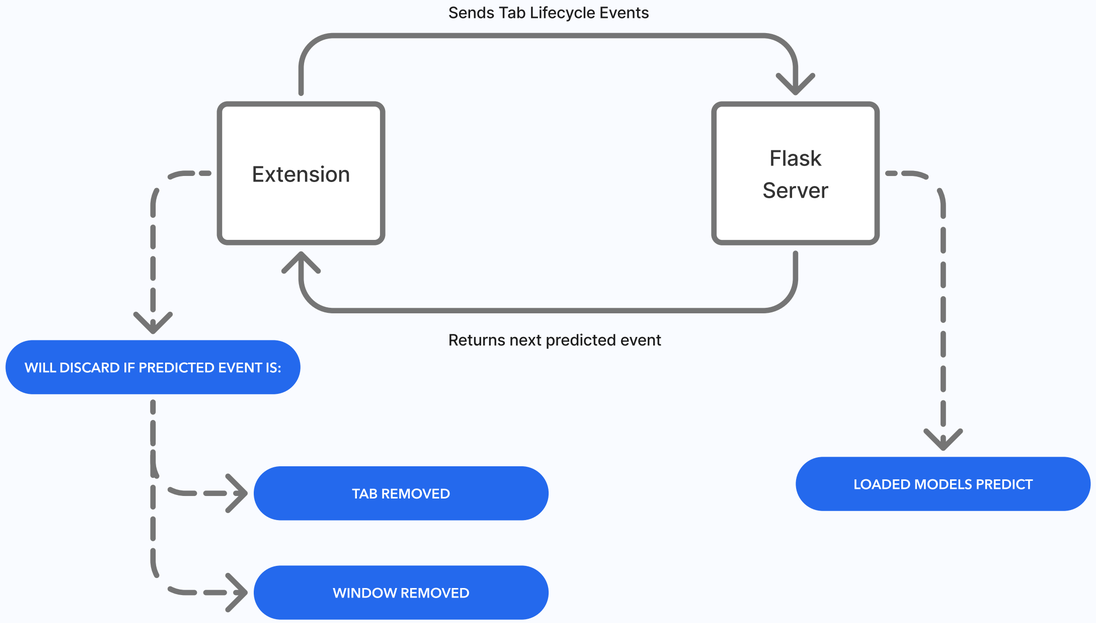
Event Based

Excessive calls

Timer based

User Configurable & Offers more control

1 , 2 , 5 , 10 , 30 Minutes ...



Results

Average of 60 - 65% of Accuracy in Predicting the correct Tab event as per the model?

Work in Progress:

False Positives

Total Memory Savings

Discussion

Information within URLs with regards to the type of activity associated with particular tabs was not leveraged.

Resource Usage metrics were excluded - not giving enough information on user behaviour and intent

Need for proper handling mechanism to cater strained hangs.

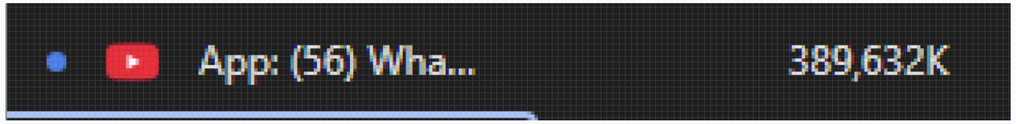
Training Data worth of 3 Days intensive usage of a single user.

Latency and resource overhead?

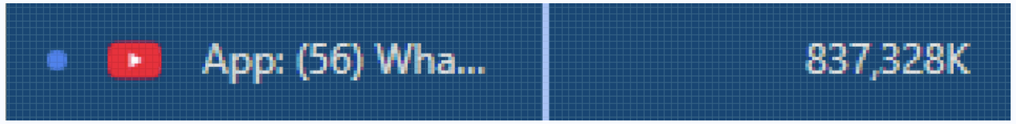
Is it light enough?

Extensions can cause additional hidden memory costs - take example of the ad blocker.

What could contribute to memory consumption?



Memory Usage Without any extensions



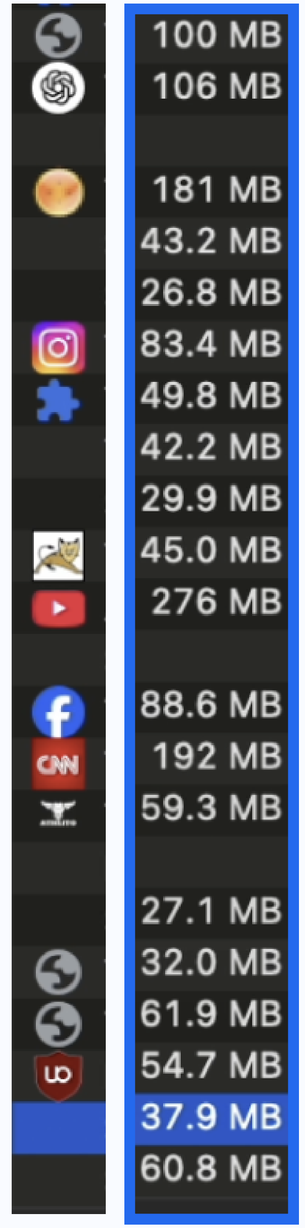
Memory Usage with extensions

Consumes approximately **~40 MB**

- 15 minute long rigorous activity for more than 10 tabs.
- For 25+ tabs and constant activity the consumption maxed at 45 MB.

Average RAM consumption of Websites

Task	Memory footprint	CPU	Network	Process ID
GPU Process	156 MB	1.7	0	5545
Utility: Network Service	61.2 MB	0.6	0	5546
Utility: Storage Service	24.5 MB	0.0	0	5547
Utility: Audio Service	21.3 MB	0.7	0	7609
Utility: Video Capture	22.3 MB	0.0	0	7610
Spare renderer	20.9 MB	0.0	0	25755
Spare renderer	20.9 MB	0.0	0	25758
Spare renderer	20.6 MB	0.0	0	25759
Spare renderer	20.7 MB	0.0	0	25760
Spare renderer	20.7 MB	0.0	0	25761
Tab: DevTools	100 MB	0.0	0	24460
Tab: Dynamic Tab Management Model	106 MB	0.0	0	24506
Dedicated worker: blob:https://chatgpt.com/131f6543-4882-42a5-8d46-				
Tab: Goku Black | Dragon Ball Wiki | Fandom	181 MB	0.2	0	25675
Subframe: https://discord.com/	43.2 MB	1.5	0	25750
Subframe: https://imrworldwide.com/	26.8 MB	0.0	0	25751
Tab: Instagram	83.4 MB	0.0	0	25757
Tab: Extensions	49.8 MB	0.0	0	25637
Tab: Rigour Maths - Success for all in Numeracy and Mathematics	42.2 MB	0.0	0	24525
Subframe: https://www.facebook.com/	29.9 MB	0.0	0	24522
Tab: Login Required - Sakai	45.0 MB	0.0	0	25636
App: (21) #padel #metaglasses #padel_by_eugene - YouTube	276 MB	3.4	0	24550
Service Worker: https://www.youtube.com/sw.js				
Tab: 3D model of Suzuki Khyber | By Suzuki Khyber and Cultus fan club | Facebook	88.6 MB	0.0	0	25635
Tab: Inside NATO chief Mark Rutte's charm offensive on Trump that shocked as much as	192 MB	0.2	0	24937
Tab: Products – ATHLITO WORLD INTL	59.3 MB	0.0	0	25026
Dedicated worker: https://www.athlito.online/				
Subframe: https://www.athlito.online/	27.1 MB	0.0	0	25027
Tab: Oracle PeopleSoft Sign-in	32.0 MB	0.0	0	25062
Tab: Discards	61.9 MB	0.0	0	25560
Extension: uBlock Origin	54.7 MB	0.0	0	25756
Service Worker: chrome-extension://jndafigkneoiec[illegible]mgfdlamnkdcbiaeh/background.js	37.9 MB	0.0	0	25615
Service Worker: chrome-extension://epcnnfbjfcgphgdmggkamkmgojdagdnn/assets/	60.8 MB	0.0	0	25700



25 MB ~ 300 MB (With add blockers)

Extension load lies in the lowest tier of tab consumption.

Possible Improvements

The model does not make use of the total number of tabs + all tabs are catered individually

Predicting tab removed can still leave much room for inactivity - potential discards → Memory Savings

How much inactivity is affordable?

Adding overall System Pressure context.

More informed discards under 1o minutes w/o compromising on user satisfaction.

Future Model?

Incorporating a model which can estimate probabilities of all the tabs likely to be visited next.

Group tabs in terms of their usage

If a particular website has the least probability of being visited across all the tabs in the frequent group - potential discard.

Cater False Positives

Conclusion

Given low system resources, high multi-tasking in browser - how do we reduce memory usage

Lightweight extension acts as an proactive measure not keep tabs that are to be removed.

Reduces unnecessary system usage at the cost of smooth usage.

Transformers can store long term dependencies and offers detail insights with regards to interpretability of model's decision

Model hosted at server will not cause load at the user's end

Reinforcement Learning Strategies can also improve learning over time with updates on the go.

Additional process running in the browser.

Thank You

Muhammad Qasim Atiq Ullah