# Chromium GSoC Contributor Proposal Template

**Summary**
self link:
Chromium GSoC 2025: Farfetc...

Add support to farfetchd for tracing and replay.

**Owner:** murphytian06@gmail.com
**Contributors:** Murphy Tian
**Approver:**
*sarthakkukreti@chromium.org*,
*asavery@chromium.org*
**Status**: **For Review**
**Created:** 2025/03/31

## Project Abstract

*(1-2 paragraphs) What are we doing and why? What problem are you trying to solve? What are the goals and NON-goals? Please make the objective understandable for someone unfamiliar with this project by including the necessary context, but keep it short. Elaborate on the details below in the Background and Requirements sections.*

*Preloading files into memory is a useful mechanism for improving application cold start performance. On ChromiumOS, use `ureadahead` to speed up boot times. Farfetchd is a general purpose D-Bus service that allows prefetch on application binaries and associated resources and allows the optimization of cold start times.*

*This project is aiming to add support to farfetchd for tracing file pages that are fetched for an application during a given workload and standardized tracepoints. And use the collected trace, preload the specified pages from disk and measure improvements in application performance. Then, add tast test which showcased chromium improvement in startup time.*

*Non-goals: integrating service for all ChromiumOS applications*

## Background

*(2 paragraphs max.) Share your understanding What background context is necessary? Try to use links or references to external sources (other docs or wikipedia), rather than writing your own explanations. Please include document titles so they can be found when links go bad. Include a glossary if necessary.*

*Note: this is background; do not write about your design, specific requirements details, or ideas to solve problems here.*

*Application cold start performance is often hindered by disk access latency, especially on resource-constrained devices. To mitigate this, operating systems can prefetch expected data into memory. Farfetchd aims to address this challenge for ChromiumOS.*

*The linux kernel offers system call tracing via tracefs, allowing us to observe calls like `sys_read`, `uselib` that load parts of an application into the memory. This enables capturing a workload's disk access pattern and replaying it ahead of time. The `mmap` syscall is used to map files into memory, facilitating efficient prefetching of disk pages. This tracing and replay mechanism is central to farfetchd to application acceleration.*

## Design Ideas

---

*(2-5 paragraphs) A short and sweet overview of your implementation ideas. If you have multiple solutions to a problem you would like feedback on, list them concisely in a short bullet list.*

*Use a diagram when necessary. Cover major structural elements in a very succinct manner. Which technologies will you use? What new components will you write? What technologies will you use to write them? What are the dominant scaling parameters? (data sizes, qps estimates, etc.) Consider the range and maximum values. What is the general rollout strategy? What are your information security concerns and how will you address them? How are you putting yourself in a position to own the successful delivery of the full solution?*

*Tracing API:*

- *Hook into `tracefs` to log system calls relevant to disk I/O*
- *Record memory page-level disk reads during an application's execution*
- *Tag each trace with a unique application identifier to allow later correlation during replay*
- *Store this trace in a trace file in a structured trace file*
- *Use appropriate filters and trace points to limit noise and ensure relevant data is captured*

*Replay API:*

- *Implement a D-Bus interface to accept application identifiers and trigger prefetch.*
- *Read the trace file and mmap() the corresponding file offsets into memory*
- *Ensure idempotent and secure replay with validation replay inputs and checking system invariants*
- *As part of validation, instrument the application's startup latency and compare against baseline cold start performance to detect regressions or improvements*

*Rollout Strategy*

- *Begin with support for a test to validate tracing and replay behavior*
- *Gradually extend to a broader set in configuration*
- *Ensure sandboxing and memory limits are respected*

*Technologies: C++, Linux Kernel (tracing subsystems), Golang (for tests)*

## Code Affected

Code primarily resides in
[https://source.chromium.org/chromiumos/chromiumos/codesearch/+/main:src/platform2/farfetchd/](https://source.chromium.org/chromiumos/chromiumos/codesearch/+/main:src/platform2/farfetchd/)

## Alternatives Considered (Optional)

*(2 paragraphs) Include alternate design ideas here which you are leaning away from.*

## Related Work

*(1-2 paragraphs) Show that you did an extensive survey about the state of the art. Gold standard: provide a table which compares your proposal w.r.t. features & limitations to existing or similar solutions.*

*Prefetching techniques are widely used across operating systems to improve application startup performance. For example, Android's `dexopt` or iOS's `Jetsam` offer similar cold-start optimization mechanisms. Unlike them, Farfetch is transparent to the app and relies solely on system-level tracing and replay.*

*A particularly relevant system is `ureadahead`, used by Ubuntu and still present in ChromiumOS. It preloads file pages at boot based on previous boot's access patterns. While effective, ureadahead focuses on system boot, not on-demand application startup. In contrast, Farfetchd aims to generalize this behavior per-application, using runtime traces and replay during app launches.*

*Comparison table*

| *System* | *Approach* | *Page-level Granularity* | *Dynamic Trace Support* |
|---|---|---|---|
| *Android dexopt* | *Compile-time optimization* | *No* | *No* |
| *iOS Jetsam* | *Heuristic memory purge* | *No* | *No* |
| *ChromiumOS Farfetchd* | *System level tracing and replay* | *Yes* | *Yes* |
| *ureadahead* | *Boot-time trace replay* | *Yes* | *Yes* |

## Pre proposal Work

*Share your starter bugs and understanding of the project*
*I solve two starter bugs and submit two PR*
*Add minijail invocation for farfetchd*
*https://chromium-review.googlesource.com/c/chromiumos/platform2/+/6395878/2*
*Add dbus activation for farfetchd*
*https://chromium-review.googlesource.com/c/chromiumos/platform2/+/6356005*

## Schedule of Deliverables (timeline)

*Select the period and dates during which you will be working on a medium or large project and plan the deliverables accordingly and if you need extended timeline to complete the work. Write a weekly plan and estimates on working on the deliverables. Please update below dates based on this year's timeline.*

### May 1 - May 26 (Community Bonding Period)

- Get familiar with Farfetchd's architecture and tracefs APIs
- Meet with mentors to finalize tracing targets
- Read ChromiumOS D-bus and tast test framework docs

### May 27 - July 12 (Phase I)

- Coding officially begins!

Week 1-2

- Implement basic tracing infrastructure using `tracefs`
- Record system calls and page-level access for a test app
- Integrate application identifier tagging
- Validate correctness via manual trace inspection

Week 3-4

- Create structured tracefile format and serialization logic
- Write simple replay logic to mmap() pages based on trace
- Begin writing CLI utilities for trace/replay

Week 5-6

- Add d-bus interface for launching replay from application ID
- Build test harness with simple tast benchmark
- Measure and log application cold start latency

Week 7

- Submit Phase evaluation
- Demonstrate end-to-end test for one application with trace + replay
- Document API and usage instructions

### July 12 - August 19 (Phase II)

- Work Period | GSoC contributors work on their project with guidance from Mentor

Week 8-9

- Add additional filters and tracing enhancements
- Validate replay behavior under failure scenarios
- Ensure sandboxing and memory limit compliance

Week 10-11

- Add support for storing multiple trace files
- Integrate metrics reporting into tast test
- Tune performance and memory overhead

Week 12

- Submit final evaluation
- Finalize tast test, document performance results, and trace format

### August 19 - November 11 (For Extended Timelines)

- GSoC contributors with extended timelines continue coding

## Communications

---

Communication is the most essential for successful GSoC. Please tell us about your work schedule, communication preferences. Tell us about how you plan on handling delays and how you will maintain regular contact with the mentor on deliverables and make steady progress.

- Timezone/ Working hours: EST 10 pm-11 am
- Email: murphy.tian@mail.utoronto.ca
- Phone +86 13104058206
- Slack id: Murphy Tian

## About Me

---

Add a section below about your background, experiences, interests. You can add Resume here.
I'm a fourth-year computer science student at University of Toronto. I am currently a research intern at Microsoft Research Asia. I was a software engineer intern in Microsoft in the Edge Mobile team.
My personal interest in computer science is constructing complex operating system, distributed system, and software engineering systems. Here is my resume link: sde_resume.pdf

## Prior Experience with open source

*Add links to all open source contributions*

1. *Social Needs Marketplace - Impact ([csse-uoft/SNM-I]) Create a calendar in the website including front-end and back-end functionality*
2. *OpenManus ([GitHub - OpenManus/OpenManus-RL: A live stream development of RL tunning for LLM agents]) one of the core contributor, be responsible for the rollout-trajectory part*
3. *libCacheSim ([GitHub - 1a1a11a/libCacheSim: a high performance library for building cache simulators]) Add a js binding for this C/C++ project*

## Why Chromium?

*Tell us about why you choose Chromium org in Google Summer of Code. How the organization can help you to reach your goals.*

*In the summer of my third year, I took an operating systems course at the University of Toronto, where I independently implemented major components of the Pintos OS from scratch—including thread management, user programs, virtual memory, and file systems. It was a demanding project that took over 300 hours, and although some parts would be easier for me now, it was a formative experience that sparked my deep interest in operating systems.*

*That same summer, I also proposed an idea and experimented with improvements to the page replacement algorithm under the guidance of my professor. While I didn't end up with a better algorithm, the process significantly deepened my understanding of memory management and reinforced my passion for low-level system work.*

*In September, I joined Microsoft as a software engineering intern on the Microsoft Edge Mobile team, which is built on Chromium. I focused on front-end development for a cross-platform UI library (for iOS, Android, and hybrid environments), with particular responsibility for the hybrid implementation using JavaScript. The library was similar in nature to FluentUI, but customized for the Edge Mobile environment. This role gave me hands-on experience working with the Chromium codebase.*

*These experiences—my growing fascination with operating systems and my direct involvement with Chromium through industry—make the Farfetch'd project especially exciting for me. It represents an opportunity to contribute to a real, large-scale operating system (ChromiumOS) and build something meaningful that impacts users at the systems level. Contributing to Chromium through GSoC would be a major milestone in my journey, boosting both my confidence and commitment to a future in systems development.*

## Feedback from Chromium

*If you read this document please provide your short general feedback in the section below. Please also feel free to make comments above.*

| Username | Date | Comment |
|---|---|---|
| | | |
| | | |