IPFS

❖امیرعلی فدیشه ای

❖سید حسام الدین دلبری

معرفی پروژه:

این پروژه یک سیستم ذخیره‌سازی و بارگذاری توزیع‌شده محتوا-محور و غیرمتمرکز برای فایل ها فراهم می کند. هدف اصلی، ایجاد یک سرویس ذخیره‌سازی و بازیابی فایل است که فایل‌ها را به صورت خودکار به قطعات کوچک یا همان چانک ها تقسیم می کند. سپس هر قطعه را با یک شناسه یکتا (هش) ذخیره می‌کند و امکان بازیابی فایل با حفظ ترتیب درست داده‌ها و یکپارچگی آنها (بر اساس محتوای آنها و نه مکان ذخیره شدن آنها) را فراهم می کند. در پایان هم بر پایه ی این هش ها یک شناسه‌ی یکتا برای کل فایل به منظور فراهم کردن امکان بازیابی و دانلود آن ساخته می شود. یکی از اهداف دیگر این پروژه جلوگیری از ذخیره ی چند باره و تکراری داده های یکسان هست.

به این صورت که داده های یکسان تنها یک بار ذخیره می شوند. همچنین تمرین مفاهیمی از سیستم‌عامل که در نیمه ی نخست ترم آموختیم، مانندِ فرآیندها، نخ‌ها(ایجاد و کار با آنها)، مدیریت همزمانی فرآیند ها و ارتباط میان آنها از اهداف آموزشی این پروژه بوده.

قابلیت‌ها و ویژگی های اصلی پروژه:

- تقسیم فایل به چانک های با اندازه ی ثابت (با قابلیت تغییر)
- پردازش موازی چانک‌ها با بهره گیری از تِرد پول
- محاسبه‌ی هش هر چانک با الگوریتم بِلِیک سه با قابلیت تغییر
- معماری دو لایه
- لایه ی موتور و لایه ی گیت ورودی
- ساخت و استفاده از شناسه ی یکتا
- ساخت شناسه ی یکتا برای هر فایل بر اساس مانیفست آن و امکان بازیابی فایل تنها توسط آن.
- مدیریت همزمانی پیشرفته
- برای پردازش موازی 4 کارگر ثابت و مدیریت کارها در صف کار
- از قفل ها و متغییر های شرطی برای هماهنگی آنها و مدریت همزمانی چندین دانلود و آپلود، بهره می بریم.
- ذخیره‌سازی اتمیک و ایمن

- نوشتن اتمیک مانیفیست، تائید هش در هر خواندن و نوشتن و جلوگیری از حذف داده های مشترک
- جلوگیری از ذخیره ی داده های تکراری
- مدیریت خطای جامع
- کدهای خطای استاندارد و پیام های خطای استاندارد
- بهینه‌سازی عملکرد
- پردازش موازی چانک‌ها، حداقل سربار ارتباطی و استفاده از الگوریتم هش سریع
- راستی‌آزمایی هر چانک هنگام دانلود

معماری کلی سیستم

معماری این سیستم از سه بخش اصلی کلاینت یا همان واسط کاربری، گیت وِی و موتور C تشکیل شده است.

Gateway (Python)

دروازه ی ورودی در این پروژه یک سرور (HTTP) مبتنی بر است که به عنوان واسط میان کاربر ThreadingHTTPServer و موتور یا هسته

ی پردازشی عمل می کند. این بخش مسئول دریافت درخواست‌های (HTTP)، مدیریت جریان داده ، و تبدیل آن‌ها به پیام‌های باینری مطابق پروتکل تعریف‌شده برای موتور پردازشی هست. این بخش هیچ گونه محاسبات و منطق سنگینی را انجام نمی دهد و مدیریت نمی کند.

ویژگی‌ها و وظایف:

- پیاده‌سازی یک سرور چندنخی با استفاده از ThreadingHTTPServer
- پشتیبانی از پردازش و انجام همزمان چند درخواست آپلود و دانلود
- سرویس‌دهی فایل‌های رابط گرافیکی

(HTML, CSS, JavaScript)

- تبدیل درخواست‌های (HTTP)، به نمایش‌های باینری قابل فهم برای موتور پردازشی
- برقراری ارتباط محلی با موتور پردازشی از طریق: Unix Domain Socket

Engine (C)

موتور و هسته‌ی اصلی و پردازشیِ سیستم ذخیره‌سازی و دانلود است که منطق مرکزی پروژه را پیاده‌سازی می‌کند. این بخش به زبان c نوشته شده و مسئول انجام همه ی عملیات‌های سنگین شامل مدیریت فایل‌ها، پردازش چانک‌ها، هش‌کردن چانک ها،ساخت شناسه ی یکتا برای هر فایل(CID)، ساخت مانیفست، و دانلود درست و ایمن فایل‌ها می‌باشد و به‌صورت یک سرویس محلی اجرا شده و از طریق Unix Domain Socket با دروازه ی ورودی ارتباط برقرار می کند این طراحی باعث جداسازی کامل منطق پردازشی از لایه‌ی ارتباطی شده و امکان مدیریت همزمانی و منابع سیستم‌عامل را به‌صورت شفاف فراهم می‌کند.

ویژگی‌ها و وظایف:

- پذیرش اتصال‌های ورودی دروازه ورودی و مدیریت هر اتصال در یک ترد مجزا
- تقسیم وظایف سنگین (هش‌کردن، ذخیره‌سازی، خواندن و راستی‌آزمایی) بین نخ‌ها
- پیاده‌سازی ذخیره‌سازی محتوامحور
- محاسبه‌ی هش هر چانک با الگوریتم

- محاسبه‌ی هش هر چانک با الگوریتم BLAKE3
- جلوگیری از ذخیره‌سازی داده‌ی تکراری با استفاده از بررسی وجود چانک و بلاک مشابه
- استفاده از thread pool برای پردازش موازی چانک‌ها
- ساخت فایل مانیفست JSON
- بارگذاری مانیفست و بازسازی درست فایل در زمان دانلود
- راستی‌آزمایی درستی هر چانک هنگام دانلود پیش از ارسال
- تضمین ترتیب صحیح چانک‌ها در خروجی دانلود
- مدیریت همزمانی با استفاده از MUTEX LOCK و متغییر شرطی
- انجام عملیات نوشتن به صورت ATOMIC برای جلوگیری از خراب شدن داده‌ها

User Interface

رابط گرافیکی پروژه یک واسط کاربری تحت وب است که برای تعامل ساده‌تر کاربر با سیستم ذخیره‌سازی طراحی شده است. این بخش به‌صورت کامل در سمت دروازه ورودی پیاده‌سازی شده و از طریق مرورگر وب در دسترس قرار می‌گیرد. هدف از ایجاد رابط گرافیکی،

فراهم‌کردن امکان آپلود و دانلود فایل بدون نیاز به ابزارهای خط فرمان و نمایش عملی عملکرد سیستم در محیطی کاربرپسند می‌باشد. رابط گرافیکی هیچ‌گونه منطق پردازشی یا ذخیره‌سازی انجام نمی‌دهد و صرفاً نقش لایه‌ی نمایشی را ایفا می‌کند؛ تمامی عملیات‌های اصلی همچنان در موتور پردازشی انجام می‌شوند.

ویژگی‌ها و وظایف:

- پیاده‌سازی رابط کاربری با استفاده از **CSS,JavaScript،HTML**
- ارائه‌ی یک صفحه‌ی وب برای تعامل مستقیم کاربر با سیستم
- امکان انتخاب فایل از سیستم کاربر
- پشتیبانی از درگ اند دراپ برای آپلود فایل
- ارسال فایل به دروازه ورودی از طریق endpoint /upload
- نمایش سی آی دی تولیدشده پس از آپلود موفق
- امکان کپی‌کردن سی آی دی و و استفاده‌ی مستقیم از آن برای دانلود
- دریافت سی آی دی از کاربر و ارسال درخواست دانلود
- دریافت فایل از طریق endpoint /download?cid=...

- نمایش وضعیت عملیات (در حال آپلود، خطا، تکمیل موفق) به کاربر

نحوه ارتباط Gateway (Python) و Engine (C)

ارتباط میان دو فایل از طریق دو مؤلفه‌ی کلیدی انجام می شود:

- ❖ **مکانیسم ارتباط میان فرآیندی:** سوکت دامنه‌ی یونیکس
- ❖ **پروتکل داده :**فِرِیم‌بندی باینری.

سوکت دامنه ی یونیکس:

یک روش سریع و محلی برای ارتباط دو فرآیند در یک سیستم‌عامل مبتنی بر یونیکس یا لینوکس است.

: Engine (c_engine.c)

ایجاد سوکت : با استفاده از socket(AF_UNIX,SOCK_STREAM,0) یک سوکت استریم دامنه‌ی یونیکس ایجاد می‌کند.

اتصال به فایل :مسیر سوکت تعریف‌شده به عنوان g_sock_path، با تابع bind به سوکت متصل می‌شود. پیش از آن، unlink فایل احتمالی باقی‌مانده از اجرای قبلی را حذف می کند.

گوش‌دادن و پذیرش :این بخش با رفتن به حالت listen آماده‌ی پذیرش اتصال می‌شود و سپس با accept هر اتصال ورودی از Gatewayرا می‌پذیرد.

همزمانی :مدیریت هر اتصال پذیرفته‌شده توسط یک **نخ مجزا** انجام می‌شود تا Engine بتواند به‌طور همزمان به چندین درخواست Gatewayپاسخ دهد. (البته شمار این نخ ها محدود هست.)

: Gateway (main.py)

اتصال :در تابع engine_call، Gateway یک سوکت جدید با socket.socket(socket.AF_UNIX, socket.SOCK_STREAM) ایجاد کرده و با s.connect(ENGINE_SOCK) به مسیر تعریف‌شده توسط Engine متصل می‌شود.

ارسال داده : پس از فریم‌بندی، همه ی پیام‌های درخواستی را با استفاده از s.sendall(m) به Engine ارسال می‌کند.

راه اندازی پروژه

قبل از اجرای پروژه، موارد زیر باید روی سیستم آماده شده باشند:

- سیستم‌عامل مبتنی بر Linux
- کامپایلر c ترجیحا استفاده از gcc
- Python نسخه 3.8 یا بالاتر
- دسترسی به ترمینال
- دانلود فایل های پروژه از ریشه اصلی ریپازیتوری پروژه در گیت

❖ Command for compile code (one-line and on root of project):

- gcc -O2 -pthread -o c_engine -DBLAKE3_NO_AVX512 -Iblake3 c_engine.c blake3/blake3.c blake3/blake3_dispatch.c blake3/blake3_portable.c blake3/blake3_sse2.c blake3/blake3_sse41.c blake3/blake3_avx2.c -msse2 -mssse3 -msse4.1 -mavx2 -ljansson

❖ run server and gateway:

- ./c_engine /tmp/cengine.sock
  python3 main.py

❖ Open your browser and enter url:

- http://127.0.0.1:9000 on your browser