دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

پروژه کارشناسی
مهندسی کامپیوتر

محاسبه فاصله استفاده مجدد

نگارش

علی مجیدی، آرتین برقی، امیر رضا اینانلو، بردیا رضائی کلانتری

استاد راهنما

حسین اسدی

تیر ۱۴۰۳

چکیده

در این مقاله، به بررسی روش‌های محاسبه فاصله استفاده مجدد در سیستم‌های حافظه نهان پرداخته شده است. فاصله استفاده مجدد معیاری است که تعداد مراجع حافظه متمایزی که بین دو مرجع به یک آدرس حافظه خاص رخ داده است را اندازه‌گیری می‌کند. این معیار برای تحلیل رفتار حافظه نهان و بهینه‌سازی عملکرد آن بسیار مهم است.

دو روش اصلی برای محاسبه فاصله استفاده مجدد معرفی شده‌اند: روش مبتنی بر استک و روش مبتنی بر درخت. روش مبتنی بر استک، که به عنوان روش ساده و ناکارآمد شناخته می‌شود، از یک استک برای ذخیره آدرس‌های مشاهده شده استفاده می‌کند و با هر بار مشاهده یک آدرس جدید، استک را جستجو کرده و عملیات‌های پاپ و پوش را انجام می‌دهد. این روش با وجود سادگی، زمان اجرایی $O(n^۲)$ دارد و برای ورودی‌های بزرگ مناسب نیست.

در مقابل، روش مبتنی بر درخت با استفاده از یک ساختار درختی بهینه‌سازی شده، فاصله استفاده مجدد را با کارآیی بیشتری محاسبه می‌کند. در این روش، هر گره درخت حاوی آدرس حافظه و تعداد گره‌های زیرین خود است و با استفاده از ترفندهای خاص، فاصله استفاده مجدد با زمان اجرایی $O(nlogn)$محاسبه می‌شود. این روش، با حفظ ترتیب آدرس‌ها در درخت به صورت مشابه با استک، علاوه بر افزایش کارآیی، دقت بالاتری نیز ارائه می‌دهد.

نتایج به‌دست‌آمده از این مطالعه نشان می‌دهد که استفاده از ساختارهای داده‌ای پیشرفته‌تر می‌تواند بهبود قابل توجهی در محاسبه فاصله استفاده مجدد و در نتیجه بهینه‌سازی عملکرد حافظه نهان داشته باشد. در نهایت، این مقاله با ارائه مقایسه‌ای جامع بین دو روش، مزایا و معایب هر کدام را مورد بررسی قرار داده و پیشنهادهایی برای تحقیقات آینده در زمینه بهینه‌سازی سیستم‌های حافظه نهان ارائه می‌دهد.

فهرست مطالب

فصل ۱

مقدمه

نخستین فصل این پروژه به معرفی مسئله، بیان اهمیت موضوع، ادبیات موضوع، اهداف پژوهش و معرفی ساختار این مقاله می‌پردازد.

۱-۱ تعریف مسئله

فاصله استفاده مجدد در حافظه نهان (Cache) به دوره زمانی بین دسترسی‌های مکرر به یک داده خاص در حافظه نهان اشاره دارد. این مفهوم نقش حیاتی در بهبود عملکرد سیستم‌های کامپیوتری ایفا می‌کند، زیرا حافظه نهان با دسترسی سریعتر نسبت به حافظه اصلی، سرعت اجرای برنامه‌ها را افزایش می‌دهد. فاصله استفاده مجدد کوتاه‌تر به معنای بهره‌وری بالاتر از حافظه نهان و کاهش زمان دسترسی به داده‌ها است.

۱-۲ اهمیت موضوع

اهمیت بررسی فاصله استفاده مجدد در حافظه نهان از آن جهت است که بهینه‌سازی این فاصله می‌تواند به طور قابل توجهی کارایی سیستم‌های کامپیوتری را بهبود بخشد. با توجه به افزایش حجم داده‌ها و پیچیدگی برنامه‌ها، استفاده بهینه از حافظه نهان می‌تواند تأثیر بسزایی در کاهش تأخیرات و افزایش سرعت پردازش داشته باشد. این موضوع نه تنها برای کاربران عادی، بلکه برای مراکز داده و سیستم‌های بزرگ مقیاس که نیاز به پردازش سریع دارند، اهمیت ویژه‌ای دارد.

۱-۳ ادبیات موضوع

تحقیقات زیادی در زمینه بهینه‌سازی حافظه نهان و بررسی الگوهای دسترسی به داده‌ها صورت گرفته است. مطالعه‌هایی نشان داده‌اند که با تحلیل الگوهای دسترسی و پیش‌بینی رفتار برنامه‌ها می‌توان فاصله استفاده مجدد را کاهش داد. الگوریتم‌های مختلف جایگزینی حافظه نهان نیز برای بهینه‌سازی این فاصله معرفی شده‌اند. برخی از این تحقیقات بر روی بهبود پیش‌بینی‌ها و برخی دیگر بر بهینه‌سازی ساختارهای حافظه نهان تمرکز داشته‌اند.

۱-۴ اهداف پژوهش

هدف اصلی این پژوهش بررسی و تحلیل فاصله استفاده مجدد در حافظه نهان با تمرکز بر بهینه‌سازی الگوهای دسترسی و کاهش این فاصله است. اهداف خاص این پژوهش شامل:

- تحلیل الگوریتم های محاسبه فاصله استفاده مجدد در حافظه نهان از نظر زمان و حافظه
- توسعه و ارزیابی الگوریتم های بهینه محاسبه فاصله مجدد در حافظه نهان.
- تحلیل محلیت برنامه بر اساس نتایج فاصله استفاده مجدد.

۱-۵ ساختار پژوهش

این پایان‌نامه در پنج فصل به شرح زیر ارائه می‌شود. فصل دوم به بیان مفاهیم اولیه می‌پردازد. فصل سوم به بررسی کارهای پیشین درزمینه محاسبه فاصله مجدد می‌پردازد. در فصل چهارم، نتایج جدیدی که در این پژوهش به‌دست آمده است، ارائه می‌شود. فصل پنجم نیز به تحلیل جمع‌بندی کارهای انجام شده در این پژوهش می پردازد.

فصل ۲

مفاهیم اولیه

در این فصل به توضیح و معرفی مفاهیمی که در انجام این پروژه مورد استفاده قرار گرفته‌اند از جمله Spatial Locality ، Temporal Locality ، Reuse Distance ، Cache و Cache policy می پردازیم.

۲-۱ locality

۲-۱-۱ محلیت زمانی(Temporal Locality)

محلیت زمانی یا Temporal Locality، یکی از اصول مهم در طراحی حافظه‌های کامپیوتری است که بیان می کند اگر یک مکان حافظه در یک زمان خاص مورد دسترسی قرار گیرد، احتمال دسترسی مجدد به همان مکان حافظه در زمان‌های نزدیک به آن، بالا است. به عبارت دیگر، داده‌ها یا دستورالعمل‌هایی که اخیرا مورد استفاده قرار گرفته‌اند، احتمال بیشتری دارند که در آینده نزدیک دوباره مورد استفاده قرار گیرند. این اصل معمولا در کش حافظه‌ها و مکانیزم‌های مدیریت حافظه استفاده می شود تا کارایی سیستم افزایش یابد.

۲-۱-۲ محلیت مکانی(Spatical Locality)

محلیّت مکانی یا (Spatial Localilty)، دیگر اصل مهم در طراحی حافظه‌های کامپیوتری است که بیان می‌کند اگر یک مکان حافظه در یک زمان خاص مورد دسترسی قرار گیرد، احتمال دسترسی

به مکان‌های حافظه نزدیک به آن مکان نیز در آینده نزدیک بالا است. به عبارت دیگر، داده‌ها یا دستورالعمل‌هایی که به مکان‌های حافظه متوالی و نزدیک به هم تعلق دارند، معمولاً به طور متوالی مورد دسترسی قرار می‌گیرند. این اصل نیز برای بهبود کارایی سیستم و بهره‌وری از کش حافظه‌ها به کار می‌رود.

۲-۲ فاصله استفاده مجدد(Reused Distance)

فاصله استفاده مجدد یا Reuse Distance، بیانگر تعداد دسترسی‌های حافظه به آدرس‌های یکتایی است که بین دو دسترسی متوالی به یک مکان حافظه خاص صورت می گیرد. به عبارت دیگر، فاصله استفاده مجدد، تعداد عملیات حافظه‌ای یکتایی است که بین دو دسترسی پیاپی به یک مکان خاص از حافظه انجام می‌شود.

۲-۲-۱ کاربردها(applications)

- بهینه‌سازی حافظه کش: تحلیل فاصله استفاده مجدد به طراحان سیستم اجازه می‌دهد تا الگوهای دسترسی به حافظه را بهتر درک کرده و کش‌ها را برای کارایی بهتر تنظیم کنند.
- پیش‌بینی کارایی حافظه: با استفاده از فاصله استفاده مجدد، می‌توان عملکرد برنامه‌ها را در سیستم‌هایی با اندازه‌های مختلف کش پیش‌بینی کرد و برنامه‌ها را بهینه‌سازی نمود.
- تخصیص و مدیریت حافظه: فاصله استفاده مجدد می‌تواند به بهینه‌سازی تخصیص حافظه و مدیریت منابع کمک کند تا اطمینان حاصل شود که داده‌های پرکاربرد در سطوح بالاتری از حافظه قرار دارند.

۲-۳ حافظه نهان(Cache)

حافظه کش یا (Cache Memory)، نوعی حافظه پرسرعت و موقت در سیستم‌های کامپیوتری است که به منظور بهبود سرعت دسترسی به داده‌ها و دستورالعمل‌ها مورد استفاده قرار می‌گیرد. کش، به عنوان یک واسط بین حافظه اصلی (RAM) و پردازنده (CPU)، عمل می‌کند و داده‌های پرکاربرد یا اخیراً استفاده شده را ذخیره می‌کند تا در صورت نیاز مجدد به آن‌ها، زمان دسترسی به حداقل برسد.

۲-۳-۱ سطوح حافظه نهان

حافظه نهان معمولاً در چند سطح (L1, L2, L3) سازماندهی می‌شود:
- سطح ۱ (L1): نزدیک‌ترین سطح به پردازنده و سریع‌ترین حافظه نهان است، اما ظرفیت کمتری دارد.
- سطح ۲ (L2): سرعت کمتری نسبت به L1 دارد ولی ظرفیت بیشتری دارد.
- سطح ۳ (L3): سرعت کمتری نسبت به L2 دارد ولی ظرفیت بسیار بیشتری دارد و بین چندین هسته پردازنده به اشتراک گذاشته می‌شود.

حافظه نهان به عنوان یکی از اجزای حیاتی در طراحی سیستم‌های کامپیوتری مدرن، نقش مهمی در بهبود عملکرد و کارایی کلی سیستم دارد.

۲-۳-۲ کاربردها(applications)

- افزایش کارایی پردازنده: با ذخیره داده‌ها و دستورالعمل‌های پرتکرار، حافظه نهان باعث کاهش زمان دسترسی پردازنده به این اطلاعات می‌شود و به این ترتیب سرعت اجرای برنامه‌ها را افزایش می‌دهد.
- کاهش تاخیر دسترسی به حافظه: حافظه اصلی (RAM) زمان دسترسی بالاتری نسبت به حافظه نهان دارد. با ذخیره موقت داده‌های پرکاربرد در حافظه نهان، نیاز به دسترسی مکرر به حافظه اصلی کاهش می‌یابد و به این ترتیب تاخیر دسترسی به حافظه کاهش می‌یابد.
- بهینه‌سازی پهنای باند حافظه: با کاهش تعداد دسترسی‌های مستقیم به حافظه اصلی، حافظه نهان به بهینه‌سازی پهنای باند حافظه کمک می‌کند و عملکرد کلی سیستم را بهبود می‌بخشد.
- افزایش بازدهی در سیستم‌های چند پردازنده‌ای: در سیستم‌های چند پردازنده‌ای، حافظه نهان به هر پردازنده اجازه می‌دهد تا به سرعت به داده‌های مورد نیاز خود دسترسی پیدا کند و از این طریق تداخل دسترسی‌ها به حافظه اصلی کاهش می‌یابد.

۲-۴ سیاست‌های حافظه نهان(Cache Policies)

سیاست‌های حافظه نهان یا (Cache Policies)، مجموعه‌ای از قوانین و الگوریتم‌ها هستند که مدیریت نحوه ذخیره‌سازی و جایگزینی داده‌ها در حافظه نهان را تعیین می‌کنند. این سیاست‌ها به منظور بهینه‌سازی کارایی حافظه نهان و بهبود عملکرد کلی سیستم مورد استفاده قرار می‌گیرند.

۲-۴-۱ انواع سیاست‌های حافظه نهان

سیاست جایگزینی(Replacement Policy)

Least Recently Used (LRU) :در این سیاست، داده‌ای که به تازگی استفاده نشده است، در صورت نیاز به فضای جدید، از حافظه نهان حذف می‌شود. این سیاست فرض می‌کند که داده‌هایی که اخیراً استفاده نشده‌اند، احتمال کمتری برای استفاده مجدد دارند.

First In, First Out (FIFO) :در این سیاست، داده‌هایی که زودتر وارد حافظه نهان شده‌اند، در صورت نیاز به فضای جدید، ابتدا از حافظه نهان حذف می‌شوند. این سیاست به ترتیب ورود داده‌ها به حافظه نهان توجه دارد.

Least Frequently Used (LFU) :در این سیاست، داده‌هایی که کمتر استفاده شده‌اند، از حافظه نهان حذف می‌شوند. این سیاست بر اساس تعداد دفعات استفاده از داده‌ها عمل می‌کند.

۲-۴-۲ کاربردهای سیاست‌های حافظه نهان

بهبود عملکرد سیستم: با استفاده از سیاست‌های مناسب، می‌توان عملکرد حافظه نهان را بهینه کرد و به این ترتیب سرعت دسترسی به داده‌ها و عملکرد کلی سیستم را افزایش داد.
افزایش کارایی کش: سیاست‌های جایگزینی موثر می‌توانند به حفظ داده‌های پرکاربرد در کش کمک کنند و نیاز به دسترسی به حافظه اصلی را کاهش دهند.
بهینه‌سازی مصرف انرژی: با کاهش تعداد دسترسی‌های غیرضروری به حافظه اصلی، مصرف انرژی سیستم نیز کاهش می‌یابد.

فصل ۳

کارهای پیشین

۳-۱ تئوری

در مقاله زیر دو روش مختلف پیاده سازی گفته شده است اولی پیاده سازی با استک است:
https://sites.cc.gatech.edu/classes/AY2013/cs7260_fall/lecture30.pdf
الگوریتم:
پردازش هر آدرس حافظه:
جستجوی آدرس در استک: هر بار که به یک آدرس حافظه برمی‌خورید، در استک به دنبال آن بگردید.
اگر آدرس در استک یافت نشد:
فاصله استفاده مجدد را بی‌نهایت (∞) در نظر بگیرید، زیرا این اولین باری است که این آدرس مشاهده شده است.
آدرس جدید را در بالای استک قرار دهید.
آدرس یافت شد: اگر آدرس در استک یافت شد:
تمام عناصر بالای آدرس مورد نظر را موقتاً از استک پاپ کنید.
آدرس مورد نظر را از استک حذف کنید و دور بیاندازید.
عناصر موقتاً پاپ شده را در ترتیب معکوس دوباره به استک فشار دهید.
فاصله استفاده مجدد را برابر با تعداد عناصری که موقتاً پاپ شده‌اند، محاسبه کنید.
آدرس جدید را در بالای استک قرار دهید.
خروجی: فاصله استفاده مجدد برای هر آدرس حافظه در جریان ورودی را محاسبه کنید. این مقادیر را می‌توان برای تولید هیستوگرام فاصله استفاده مجدد استفاده کرد.

اردر زمانی این برنامه $O(n^2)$است و برای ورودی های بزرگ مناسب نیست.
روش دوم پیاده سازی با درخت که از نظر اردر زمانی با خواسته پروژه مطابقت دارد و ما روش دوم را پیاده سازی کرده ایم که در فصل نتایج جدید به آن می پردازیم.

۳-۲ پیاده سازی

در لینک زیر کد c++ ای قرار داده شده است که در آن با استفاده از مفهوم LinkedList برنامه ای نوشته شده است که فاصله استفاده مجدد را حساب می کند.

https://drive.google.com/drive/folders/1wK6Jh80SLH5whCfKBOeW9QI7_290TkxC

اجزای اصلی برنامه :

برنامه به چند بخش اصلی تقسیم می‌شود:

تابع اصلی: شامل کدهایی برای خواندن ورودی‌ها، پردازش داده‌ها و نوشتن خروجی‌ها.

لیست پیوندی: برای نگهداری درخواست‌های حافظه و مدیریت دسترسی‌ها.

نحوه محاسبه فاصله استفاده مجدد :

خواندن و پردازش داده‌ها: داده‌ها از فایل‌های ورودی خوانده می‌شوند. فایل‌های ورودی شامل آدرس‌های حافظه و تعداد دفعات استفاده مجدد هستند.

ذخیره درخواست‌ها در لیست پیوندی: هر درخواست حافظه به لیست پیوندی اضافه می‌شود.

حذف و محاسبه فاصله استفاده مجدد: اگر آدرس حافظه در لیست پیوندی موجود باشد، حذف می‌شود و فاصله زمانی محاسبه می‌شود. همچنین آمار مربوط به تعداد و اندازه درخواست‌ها به‌روز می‌شود. ذخیره آمار در وکتور: مقادیر آماری محاسبه شده در یک وکتور ذخیره می‌شود. . تحلیل پیچیدگی زمانی و فضایی :

پیچیدگی زمانی : (Time Complexity)

خواندن داده‌ها از فایل‌ها: $O(n)$ با توجه به اینکه n تعداد خطوط فایل ورودی است.

جستجو در لیست پیوندی: در بدترین حالت $O(n)$برای هر جستجو. اضافه و حذف در لیست پیوندی: $O(1)$برای اضافه کردن و $O(n)$برای حذف کردن.

محاسبه آمار: $O(nlogn)$برای مرتب‌سازی و $O(n)$برای محاسبه مد و میانه.

بنابراین، پیچیدگی کلی زمانی $O(n^2)$ می‌باشد.

پیچیدگی فضایی : (Space Complexity)

ذخیره داده‌ها در وکتورها و لیست پیوندی: $O(n)$

ذخیره آمار در وکتور: $O(n)$

بنابراین، پیچیدگی فضایی $O(n)$می‌باشد.

به طور خلاصه این برنامه برای محاسبه فاصله استفاده مجدد از حافظه نوشته شده و از لیست پیوندی برای مدیریت درخواست‌های حافظه استفاده می‌کند. پیچیدگی زمانی آن $O(n^2)$و پیچیدگی فضایی آن $O(n)$است. برنامه با استفاده از توابع مختلف آمارهای مربوط به دسترسی‌های حافظه را محاسبه و در فایل خروجی ذخیره می‌کند.

فصل ۴

نتایج جدید

۴-۱ الگوریتم

طبق مقاله ای که در قسمت قبل به آن اشاره شده، روشی برای محاسبه فاصله استفاده مجدد با استفاده از نوعی درخت دودویی خاص معرفی شد، که پیچیدگی زمانی آن $O(nlogn)$ و حافظه مصرفی آن $O(n)$ بود.

در این الگوریتم هر node ۶ مقدار را در خود نگه می‌دارد:

۱. مقدار آدرس

۲. ارتفاع زیر درخت

۳. تعداد راس‌های زیر درخت

۴. اشاره به فرزند چپ در صورت وجود

۵. اشاره به فرزند راست در صورت وجود

۶. اشاره به پدر

به دو عدد هش مپ هم نیاز داریم:

۱. از آدرس به راس مربوطه در درخت

۲. از آدرس به لیستی از فاصله استفاده مجدد

الگوریتم:

۱: تمام آدرس‌ها را از فایل استخراج کرده و آن‌ها را به ترتیب زمان دسترسی مرتب می‌کنیم (در اینجا از ابتدا مرتب است و نیازی به این کار نیست).

۲: به ازای هر آدرس:

۲-۱: برسی می‌کنیم که آیا در هش مپ به راسی مپ شده است یا خیر اگر نشده بود، یک برگ جدید به چپِ چپ‌ترین برگ درخت اضافه می‌کنیم و مقادیر تعداد راس‌های زیر درخت را بروزرسانی می‌کنیم و همینطور با چرخش درخت را متوازن می‌کنیم. در هش مپ فاصله استفاده مجدد، آن آدرس را به یک لیست خالی مپ می‌کنیم.

۲-۲: اگر راسی وجود داشت:

۲-۲-۱: ابتدا جایگاه آن راس در inorder traversal را بدست می‌آوریم (شروع از صفر) و آن را به لیست متناظر آن آدرس اضافه می‌کنیم. برای این کار از تعداد راس‌های زیر درخت که برای هر راس ذخیره کردیم استفاده می‌کنیم.

۲-۲-۲: آن راس را حذف کرده و برگ جدیدی به چپِ چپ‌ترین برگ درخت اضافه می‌کنیم. در هر دو مرحله، مقادیر تعداد راس‌های زیر درخت را بروزرسانی می‌کنیم و با چرخش درخت را متوازن می‌کنیم. در نهایت هم راس متناظر به آن آدرس را در هش مپ بروزرسانی می‌کنیم.

۲-۳: به آدرس بعدی و مرحله ۲-۱ می‌رویم.

۳: در نهایت به ازای هر آدرس لیستی از فاصله استفاده مجدد از آن آدرس را داریم که با این اطلاعات می‌توانیم هر داده آماری را محاسبه کنیم.

سورس کد در این لینک قرار دارد.

۲-۴ پیچیدگی زمانی و حافظه‌ای

درخت استفاده شده متوازن است در نتیجه ارتفاع آن از اردر $log(n)$ است. از آنجایی که تمام عملیات‌های پیاده سازی شده برای درخت نسبت به ارتفاع درخت خطی هستند، عملیات‌ها در اردر $log(n)$ هستند.

در این الگوریتم به ازای هر آدرس تعداد مشخصی عملیات بر روی درخت انجام می‌شود و با در نظر گرفتن n آدرس، پیچیدگی زمانی الگوریتم $O(nlog(n))$ است.

در بخش حافظه هم ۲ هش مپ داریم که از اردر n هستند پس فضای اشغالی الگوریتم $O(n)$ است.

۴-۳ خروجی کد برای فایل‌های پیگیری مختلف

نتایج حاصل از اجرای کدهای نوشته شده بر روی فایل‌های پیگیری Alibaba در این لینک قرار دارد.

۴-۳-۱ خروجی A669.csv

```
access count: 27244833
distinct accesses: 179031
average access count per offset: 152.17941585535465
median access count per offset: 6
std access count per offset: 809.723703091215883204535544556423520378749257678 59
max access count per offset: 11421
count of offsets with no reuse: 15908
average reuse distances: 7292.3958530768827762798235204705923733573459230951
median reuse distances: 3153.0
std reuse distances: 19779.269581852503085076357696661465178146827586377
min reuse distances: 0
max reuse distances: 178209
time: 1166.8812279701233
```

۴-۳-۲ خروجی A129.csv

```
access count: 17712493
distinct accesses: 2999197
average access count per offset: 5.905745104439622
median access count per offset: 1
std access count per offset: 261.160546083141904134044895966055148431038931594 17
max access count per offset: 250614
count of offsets with no reuse: 1623383
average reuse distances: 235256.842837593969427380513516481962980966331405 28
median reuse distances: 25003.0
std reuse distances: 460156.985737916017248777798260386003552922049390 04
min reuse distances: 0
max reuse distances: 2973720
time: 842.1423711776733
```

۴-۳-۳ خروجی A108.csv

```
access count: 20205318
distinct accesses: 1658530
average access count per offset: 12.182666578234944
median access count per offset: 5.0
std access count per offset: 170.995952726352808781961445573792170914458649795 36
max access count per offset: 111784
count of offsets with no reuse: 372946
average reuse distances: 324007.165047338655081408166201069425067025082725 91
median reuse distances: 409722.0
std reuse distances: 240484.387867914841689737271628870056394049492249 29
min reuse distances: 0
max reuse distances: 1656075
time: 1173.313912153244
```

۴-۳-۴ خروجی A42.csv

```
access count: 5087932
distinct accesses: 154348
average access count per offset: 32.964029336304975
median access count per offset: 26.0
std access count per offset: 416.713393845549402490768106586303444624948395601 64
max access count per offset: 89662
count of offsets with no reuse: 7768
average reuse distances: 65486.807471404155680738384103726621458152937094007
median reuse distances: 73544.0
std reuse distances: 39501.648696309106539314441863791375263919561982094
min reuse distances: 0
max reuse distances: 154114
time: 275.7883653640747
```

فصل ۵

نتیجه‌گیری

۵-۱ محلیت زمانی

محلی‌سازی زمانی به نحوه دسترسی به داده‌ها در طول زمان و فاصله زمانی بین دسترسی‌های مکرر به یک داده خاص اشاره دارد. با توجه به متریک‌های ارائه شده، می‌توان تحلیل دقیق‌تری از محلی‌سازی زمانی برای هر فایل ارائه داد.

۵-۱-۱ فایل A42.csv:

- میانگین فاصله های استفاده مجدد: 65486.8
- میانه فاصله های استفاده مجدد: 73544
- انحراف معیار فاصله‌های استفاده مجدد: 39501.6
- حداقل فاصله استفاده مجدد: 0
- حداکثر فاصله استفاده مجدد: 154114

فایل A42 دارای میانگین و میانه فاصله استفاده مجدد نسبتاً بالایی است، که نشان می‌دهد دسترسی‌ها به یک داده خاص در فاصله‌های زمانی طولانی‌تر انجام می‌شوند. این موضوع می‌تواند به دلیل دسترسی به داده‌های مختلف در بازه‌های زمانی مختلف باشد. انحراف معیار بالا نیز نشان‌دهنده تنوع زیاد در فاصله‌های استفاده مجدد است.

۵-۱-۲ فایل A108.csv:

- میانگین فاصله‌های استفاده مجدد: 324007.2
- میانه فاصله‌های استفاده مجدد: 409722
- انحراف معیار فاصله‌های استفاده مجدد: 240484.4
- حداقل فاصله‌های استفاده مجدد: 0
- حداکثر فاصله‌های استفاده مجدد: 1656075

فایل A108 دارای میانگین و میانه فاصله استفاده مجدد بسیار بالایی است که نشان‌دهنده محلی‌سازی زمانی ضعیف‌تری است. دسترسی‌ها به داده‌های خاص به ندرت تکرار می‌شوند و فاصله‌های زمانی بین دسترسی‌های مکرر بسیار زیاد است. انحراف معیار بالا نیز نشان‌دهنده تنوع بسیار زیاد در فاصله‌های استفاده مجدد است.

۵-۱-۳ فایل A129.csv:

- میانگین فاصله‌های استفاده مجدد: 235256.8
- میانه فاصله های استفاده مجدد: 25003
- انحراف معیار فاصله های استفاده مجدد: 460156.9
- حداقل فاصله استفاده مجدد: 0
- حداکثر فاصله استفاده مجدد: 2973720

فایل A129 دارای میانگین فاصله استفاده مجدد نسبتاً بالاست اما میانه فاصله استفاده مجدد پایین‌تری نسبت به میانگین دارد که نشان می‌دهد بیشتر دسترسی‌ها در فاصله‌های زمانی کوتاه‌تر انجام می‌شوند، اما برخی دسترسی‌ها فاصله‌های زمانی بسیار طولانی‌تری دارند که میانگین را بالا می‌برند. انحراف معیار بسیار بالا نیز نشان‌دهنده تنوع بسیار زیاد در فاصله‌های استفاده مجدد است.

۵-۱-۴ فایل A669.csv:

- میانگین فاصله‌های استفاده مجدد: 7292.4
- میانه فاصله های استفاده مجدد: 3153
- انحراف معیار فاصله های استفاده مجدد: 19779.3
- حداقل فاصله استفاده مجدد: 0
- حداکثر فاصله استفاده مجدد: 178209

فایل A669 دارای میانگین و میانه فاصله استفاده مجدد بسیار پایینی است که نشان‌دهنده محلی‌سازی زمانی بسیار خوب است. دسترسی‌ها به داده‌های خاص به طور مکرر و در فاصله‌های زمانی کوتاه‌تر انجام می‌شوند. انحراف معیار نیز پایین‌تر است که نشان‌دهنده تنوع کمتری در فاصله‌های استفاده مجدد است.

۵-۱-۵ نتیجه گیری

- A669 دارای بهترین محلی‌سازی زمانی است. میانگین و میانه فاصله‌های استفاده مجدد پایین و انحراف معیار نسبتاً کم نشان‌دهنده دسترسی مکرر و نزدیک به هم به داده‌های خاص است.
- A42 دارای محلی‌سازی زمانی مناسبی است، اما نه به خوبی A669. فاصله‌های استفاده مجدد بالاتر نشان‌دهنده دسترسی‌های کمتر مکرر به داده‌ها است.
- A129 دارای محلی‌سازی زمانی ضعیف‌تری نسبت به A42 و A669 است. میانگین بالا و انحراف معیار بسیار زیاد نشان‌دهنده تنوع زیاد در فاصله‌های استفاده مجدد است.
- A108 دارای ضعیف‌ترین محلی‌سازی زمانی است. میانگین و میانه فاصله استفاده مجدد بسیار بالا و انحراف معیار بزرگ نشان‌دهنده دسترسی‌های بسیار پراکنده و نامنظم به داده‌ها است.

۵-۲ محلیت فضایی

برای تحلیل محلی‌سازی فضایی، لازم است که اطلاعات خاصی درباره الگوهای دسترسی به حافظه و نحوه پراکندگی داده‌ها در حافظه داشته باشیم. متریک‌هایی که ارائه شده‌اند بیشتر مربوط به محلی‌سازی زمانی هستند و نمی‌توانند به طور مستقیم محلی‌سازی فضایی را تعیین کنند. در زیر به دلایلی اشاره می‌شود که چرا این داده‌ها برای نتیجه‌گیری درباره محلی‌سازی فضایی کافی نیستند:

۱. عدم وجود اطلاعات درباره ترتیب دسترسی به داده‌ها: محلی‌سازی فضایی به نحوه دسترسی به داده‌ها در نزدیکی یکدیگر در حافظه اشاره دارد. برای تحلیل این موضوع، باید بدانیم که دسترسی به داده‌ها به ترتیب چگونه انجام می‌شود و آیا داده‌های نزدیک به یکدیگر در حافظه به صورت متوالی دسترسی پیدا می‌کنند یا خیر. داده‌های ارائه شده فقط اطلاعاتی درباره تعداد دسترسی‌ها و فاصله‌های زمانی بین دسترسی‌ها ارائه می‌دهند، نه ترتیب دقیق دسترسی‌ها.

۲. عدم وجود اطلاعات درباره موقعیت‌های فیزیکی داده‌ها: برای تحلیل محلی‌سازی فضایی، باید بدانیم که داده‌ها در کجاهای حافظه فیزیکی قرار دارند. بدون داشتن اطلاعات درباره آدرس‌های حافظه‌ای که داده‌ها در آن قرار دارند، نمی‌توان تشخیص داد که آیا دسترسی به داده‌های نزدیک به هم انجام می‌شود یا خیر. اطلاعات ارائه شده شامل موقعیت‌های فیزیکی داده‌ها در حافظه نیستند.

۳. توجه بیشتر به فاصله‌های زمانی: متریک‌های ارائه شده بیشتر بر روی فاصله‌های زمانی بین دسترسی‌های مکرر به داده‌ها تمرکز دارند، مانند میانگین و میانه فاصله‌های استفاده مجدد. این متریک‌ها بیشتر به محلی‌سازی زمانی اشاره دارند و نشان می‌دهند که چگونه دسترسی به داده‌ها در طول زمان توزیع می‌شود.

۴. عدم وجود متریک‌های محلی‌سازی فضایی: برای تحلیل محلی‌سازی فضایی، متریک‌هایی مانند نسبت دسترسی‌های متوالی به داده‌های نزدیک به هم، یا میانگین فاصله بین آدرس‌های حافظه‌ای متوالی لازم هستند. هیچ‌کدام از متریک‌های ارائه شده این نوع اطلاعات را فراهم نمی‌کنند.

۵-۲-۱ نتیجه گیری

با توجه به دلایل بالا، داده‌های ارائه شده فقط می‌توانند به تحلیل محلی‌سازی زمانی کمک کنند و نمی‌توانند نتیجه‌گیری دقیقی درباره محلی‌سازی فضایی ارائه دهند. برای تحلیل دقیق‌تر محلی‌سازی فضایی، نیاز به اطلاعات بیشتر و متریک‌های خاص‌تری داریم که نحوه پراکندگی و ترتیب دسترسی به داده‌ها در حافظه را نشان دهند.