دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
پروژه پایانی درس معماری کامپیوتر

عنوان:

محاسبه فاصله استفاده مجدد

Calculating Reuse Distance

نگارش

اشکان تاریوردی

روژین تقی‌زادگان

کیان قاسمی

امیرکسری احمدی

استاد

دکتر حسین اسدی

تیر ۱۴۰۳

فهرست مطالب

۱ ساختار داده

برای محاسبه فاصله مجدد داده ها از ساختار داده درختی استفاده شده است به این صورت که در هر مرحله ترتیب $inorder$ درخت ثابت می ماند.
هر گره در درخت علاوه بر اشاره به پدر و فرزندانش یک داده و لیبل در خود ذخیره می کند که درواقع عددی است که به $offset$ داده ی آن گره مپ شده است و درواقع وضعیت آخرین باری که این $offset$ دیده شده است را نشان می دهد. هر گره یک $rank$ نیز دارد که درواقع تعداد گره های زیر درختی است که ریشه آن گره مشخص شده است، در ادامه چگونگی اضافه کردن و حذف دیتا و همچنین محاسبه فاصله مجدد در این درخت توضیخ داده شده است:

تابع $insert$:

همانطور که بالا تر توضیح داده شد، ترتیب $inorder$ درخت ثابت می ماند و معادل ترتیب داده ها است، حال داده جدیدی که اضافه می شود باید چپ ترین گره درخت باشد و درواقع داده جدید را به عنوان فرزند سمت چپ چپ ترین گره درخت معرفی می کنیم و سپس داده های گره ها را تا ریشه در $O(logn)$ به روزرسانی می کنیم.

تابع $delete$:

در این تابع باید گره مشخص شده را از درخت حذف کنیم، برای این کار با استفاده به $right-rotate$ و $left-rotate$ و حالت بندی روی وضعیت فرزندان بدون خراب کردن ترتیب $inorder$ گره مدنظر را به وضعیت یک برگ درخت می رسانیم و سپس حذف می کنیم، در طی این فرایند مقادیر $rank$ و ارتفاع گره ها را تا ریشه به روزرسانی می کنیم.

تابع $get-rank$:

این تابع زمانی فراخانی می شود که داده ای که در حال خواندن هستیم داخل درخت باشد دقت کنید که چون خواص BST در درخت صادق نیست، یک دیکشنری کلی از $content$ یا همان داده هر گره به خود گره نگه داشتیم و همچنین گفتیم که هر $offset$ به داده و لیبل خودش مپ شده است، حال با توجه به این تعاریف در $O(۱)$ می توان می فهمید که $offset$ فعلی داخل درخت وجود دارد

یا ندارد. برای محاسبه فاصله مجدد دقت کنید که دیتای فعلی آخرین داده ای است که دیده شده است و چون ترتیب $inorder$ درخت معادل ترتیب کلی اعداد است، درواقع باید برای گره فعلی یا مشخص شده تعداد گره هایی که داده آن ها از این گره بیش تر است را پیدا کنیم.
دقت کنید که همچنین گره هایی درواقع آنهایی هستند که در ترتیب $in-order$ زود تر ظاهر می شوند، برای پیدا کردن آن ها با توجه به خاصیت $rank$ کافیست به سمت ریشه درخت حرکت کنیم و محاسبات را انجام دهیم، پس از محاسبه فاصله مجدد باید گره ی داده فعلی به روز رسانی شود و از درخت حذف شود و سپس دوباره به درخت اضافه شود که با توجه به توضیحات داده شده این کار در $O(logn)$ انجام می شود.

به طور کلی پیدا کردن فاصله مجدد در یک لیست به طول n با توجه به توضیحات داده شده نیاز به $O(n)$ حافظه دارد و مرتبه زمانی اجرای آن $O(nlogn)$ است.

۲ پیاده سازی الگوریتم

در این قسمت کد ابتدا ساختار داده درختی را که در قسمت قبل توضیح داده شد تعریف کردیم سپس یک دیکشنری به نام $last - occurrence$ در نظر میگیریم که هر $offset$ را به ترتیب دیده شدن در داده های فعلی مپ می کند و همچنین یک لیست در نظر میگریم که $reuse - distance$ های دیده شده را ذخیره می کند سپس شروع به گرفتن خط به خط فایل csv می کنیم و به $offset$ مشاهده شده نگاه می کنیم، اگر داخل دیکشنری بود یعنی تا به حال دیده شده و در این صورت به سراغ محاسبه $reuse - distance$ برای این $offset$ می رویم و اگر دیده نشده بود یک $node$ با این $offset$ به درخت اضافه می کنیم. قطعه کد این قسمت در شکل زیر قابل مشاهده است.

```
def calculate_reuse_distance(traceFile):
    tree = Tree()
    last_occurrence = {}
    reuse_distances = {"avg": 0, "min": float('inf'), "max": -1, "variance": 0, '
    list_reuse_distances = []
    index = 1
    while index <= 256 * 1024:
        if index < 1024:
            reuse_distances[str(index)] = 0
        else:
            temp = index / 1024
            reuse_distances[str(temp) + "K"] = 0
        index *= 2
    reuse_distances['>256K'] = 0
    with open(traceFile, 'r') as file:
        rows = file.readlines()
        for i, row in enumerate(rows):
            time_stamp, _, offset, _, _, _, _, _ = row.strip().split(',')
            if offset in last_occurrence:
                last_index = last_occurrence[offset]
                rank = tree.get_rank(last_index)
                list_reuse_distances.append(rank)
                tree.delete(last_index)
            last_occurrence[offset] = i
            tree.insert(i)
    convert_list_to_output(reuse_distances, list_reuse_distances)
    return reuse_distances
```

سپس پس از تمام شدن این قسمت سراغ تحلیل داده های گرفته شده می رویم و min و max و $variance$ و ... را محاسبه می کنیم و همچنین فاصله مجدد های دیده شده را به بازه های $[۲^{(x-۱)}, ۲^x - ۱]$ تقسیم می کنیم که با این کار در واقع $frequency$ داده های دیده شده را در بازه های لگاریمی در نظر میگریم، قطعه کد این قسمت در شکل زیر قابل مشاهده است.

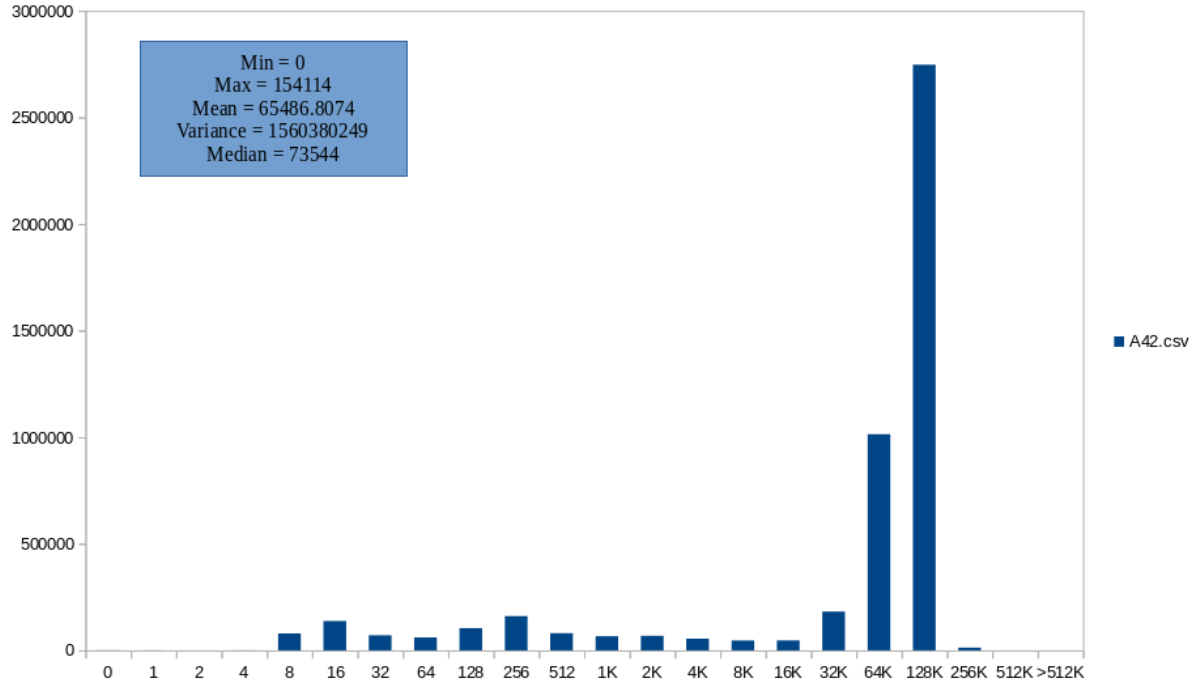
```python
def convert_list_to_output(reuse_distances, data_list):
    # reuse_distances['mode'] = most_frequent(data_list)
    data_list = np.array(data_list)
    reuse_distances['variance'] = np.std(data_list) ** 2
    reuse_distances['avg'] = np.mean(data_list)
    reuse_distances['median'] = np.median(data_list)
    for element in data_list:
        update_reuse_distance(element, reuse_distances)

def update_reuse_distance(rank, reuse_distances):
    if rank == 0:
        reuse_distances["0"] += 1
    elif rank == 1:
        reuse_distances["1"] += 1
    elif rank == 2:
        reuse_distances["2"] += 1
    elif rank < 4:
        reuse_distances["4"] += 1
    elif rank < 8:
        reuse_distances["8"] += 1
    elif rank < 16:
        reuse_distances["16"] += 1
    elif rank < 32:
        reuse_distances["32"] += 1
    elif rank < 64:
        reuse_distances["64"] += 1
    elif rank < 128:
        reuse_distances["128"] += 1
    elif rank < 256:
        reuse_distances["256"] += 1
    elif rank < 512:
```

فایل پایتون به همراه فایل csv خروجی نیز پیوست شده است.

۳ تحلیل و بررسی چهار فایل ردیابی علی‌بابا

فایل ۴۲A



شکل ۱: فایل ردیابی علی‌بابا ۴۲A

همان‌طور که مشاهده می‌شود، فاصله استفاده مجدد $۱۲۸k$ پرتکرارترین فاصله استفاده مجدد در بین سایر مقادیر است. این رخداد می‌تواند به علت دو دسترسی متوالی مانند یک خواندن پس از نوشتن یا یک نوشتن پس از خواندن باشد.

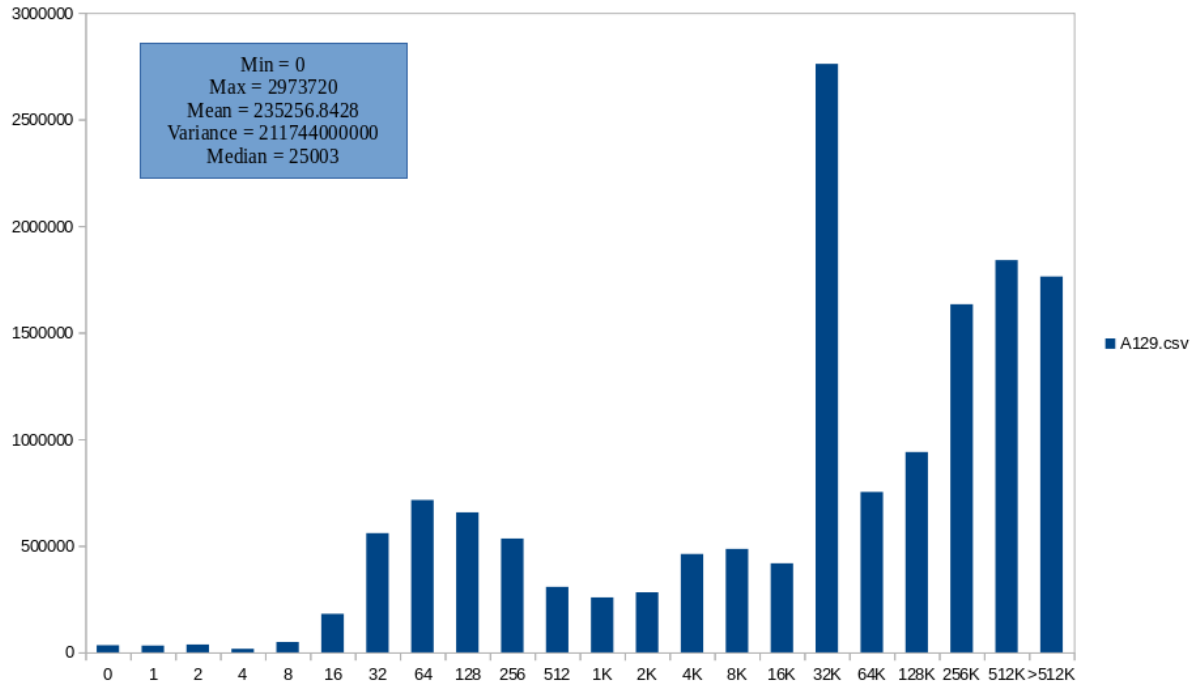
همچنین فاصله استفاده مجدد حداقل ۲۵۶هزار کم‌تکرارترین فاصله استفاده مجدد در بین سایرین و برابر با صفر مطلق است. که این رخداد نشان‌دهنده این است که بین دو دسترسی متوالی به یک خانه از حافظه، بیشتر از ۲۵۶هزار آدرس متمایز وجود ندارد.

در بازه فاصله استفاده مجدد ۰ تا ۷، تعداد داده‌ها تغییر چندانی نمی کند. در حالی که در بازه فاصله استفاده مجدد ۸ تا ۳۲ هزار یک افزایش کمی در تعداد داده‌ها رخ می‌دهد و سپس مجددا در بازه ۶۴ هزار و ۱۲۸ هزار تعداد داده‌ها افزایش ناگهانی دارد. بنابراین می‌توان نتیجه گرفت که بازه فاصله استفاده مجدد ۱۲۸ هزار می‌تواند مقدار ایده‌آلی برای فاصله استفاده مجدد در این بار کاری باشد.

طبق نتایج به دست آمده:

- کمترین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر با صفر است که بیانگر دو دسترسی پشت سر هم به یک خانه از حافظه است.
- بیشترین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۱۵۴۱۱۴ می‌باشد.
- میانگین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۶۵۴۸۶ می‌باشد. با توجه به بازه گسترده فاصله‌های استفاده مجدد، این مقدار قابل قبول است و کمتر بودن میانگین فاصله استفاده مجدد نشان‌دهنده بیشتر بودن محلیت زمانی و مکانی است زیرا در یک بازه زمانی کوتاه‌تر، دسترسی‌های بیشتری به یک خانه از خافظه انجام شده است.
- واریانس تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۱۵۶۰۳۸۰۲۴۹ می‌باشد که مقدار بسیار بزرگی است. این واریانس زیاد نشان‌دهنده این است که الگوهای دسترسی به حافظه متنوع است؛ دسترسی به برخی داده‌ها مکرر و به داده‌های دیگر کم است.
- میانه تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه ۷۳۵۴۴ می‌باشد. علی‌رغم بازه گسترده فاصله‌های استفاده مجدد، به دلیل تراکم بیشتر تعداد داده‌ها در بازه فاصله استفاده مجدد ۶۴ تا ۱۲۸ هزار، میانه در این بازه قرار گرفته است.

فایل A۱۲۹



شکل ۲: فایل ردیابی علی‌بابا A۱۲۹

همان‌طور که مشاهده می‌شود، فاصله استفاده مجدد ۳۲هزار پرتکرارترین فاصله استفاده مجدد در بین سایر مقادیر است. این رخداد می‌تواند به علت تکرر بالای دو دسترسی متوالی مانند یک خواندن پس از نوشتن یا یک نوشتن پس از خواندن باشد.

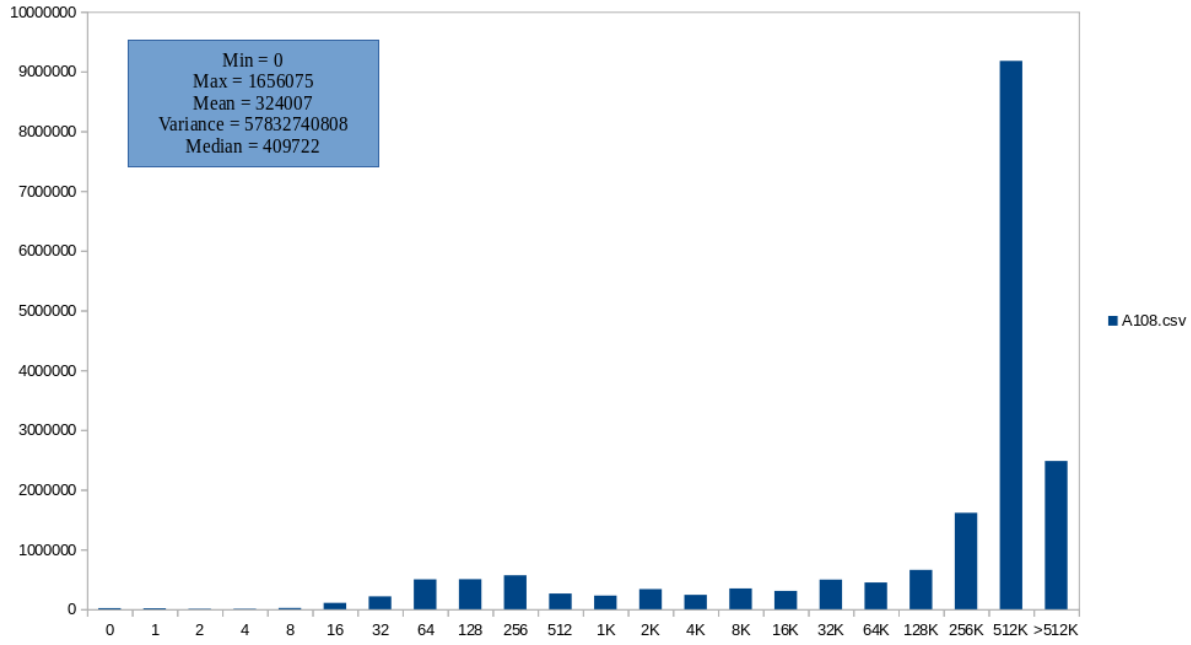
همچنین تعداد تکرار فاصله استفاده مجدد کم تر از ۸ به شدت کم است به طوری که نزدیک به صفر است. این رخداد نشان می‌دهد که بین دو دسترسی متوالی به یک خانه از حافظه، به ندرت کم تر از ۸ آدرس متمایز وجود دارد.

در بازه فاصله استفاده مجدد ۱۶ تا ۱۶هزار، تعداد داده‌ها در حال نوسان است. در حالی که در بازه فاصله استفاده مجدد ۳۲ هزار یک افزایش ناگهانی در تعداد داده‌ها رخ می‌دهد و پس از آن تعداد داده‌های کم تر است و در واقع در این بازه به بیشینه نسبی خود می‌رسید. بنابراین می‌توان نتیجه گرفت که بازه فاصله استفاده مجدد ۳۲ هزار می‌تواند مقدار ایده‌آلی برای فاصله استفاده مجدد در این بار کاری باشد چرا که دراین قسمت افزایش ناگهانی داریم و تا قبل از آن نوسانات بسیار کم است و بعد از آن نیز آن قدری تغییرات تاثیر گذار نیست.

طبق نتایج به دست آمده:

- کمترین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر با صفر است که بیانگر دو دسترسی پشت سر هم به یک خانه از حافظه است.
- بیشترین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۲۹۷۳۷۲۰ می‌باشد.
- میانگین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۲۳۵۲۵۶ می‌باشد. این مقدار بسیار بیشتر از بار کاری قبلی است که یکی از دلایل آن می‌تواند گستردگی فاصله‌های استفاده مجدد و وجود مقادیر بزرگ (نسبت به بار کاری قبلی) برای فاصله‌هاست. این مورد نشان‌دهنده این است که محلیت زمانی و مکانی در این بار کاری نسبت به بار کاری قبلی بسیار کم تر است.
- واریانس تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۲۱۱۷۴۴۰۰۰۰۰۰ است که از حالت قبلی بیشتر است. در صورتی که حافظه نهان نتواند یه گستره وسیع از داده‌ها را در خود جا دهد، واریانس بالا می‌تواند منجر به آلودگی حافظه نهان شود.
- میانه تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه ۲۵۰۰۳ می‌باشد. علی‌رغم بازه گسترده فاصله‌های استفاده مجدد، به دلیل تراکم بیشتر تعداد داده‌ها در بازه فاصله استفاده مجدد ۳۲ هزار ، میانه در این بازه قرار دارد..

فایل A۱۰۸



شکل ۳: فایل ردیابی علی‌بابا A۱۰۸

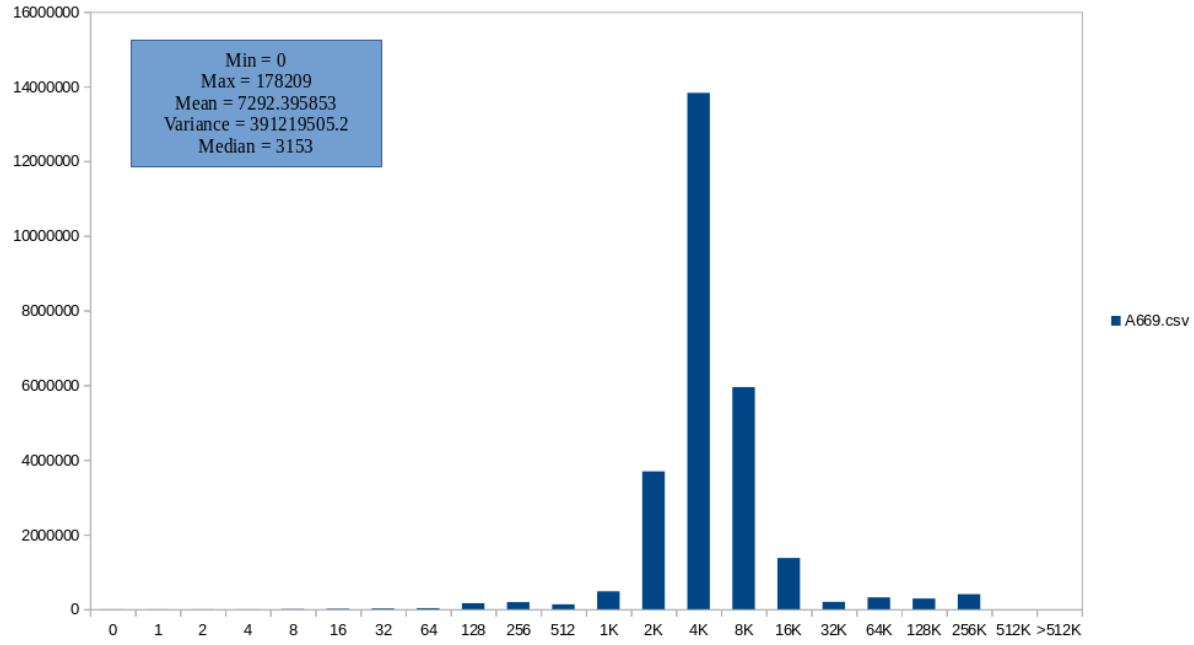
همانطور که مشاهده می‌کنید، در بازه ۰ تا ۷، تعداد داده‌ها با افزایش فاصله استفاده مجدد، تغییری پیدا نمی کند و نزدیک به صفر است و سپس در بازه ۸ تا ۱۲۸ هزار به صورت نوسانی و کوچک تغییر می کند، ام در ۲۵۶ هزار و ۵۱۲ هزار یک افزایش ناگهانی در تعداد داده‌ها رخ می‌دهد و سپس در انتها، فاصله استفاده مجدد، کاهش می یابد. بنابراین، می‌توان گفت که بازه ۵۱۲ هزار، ایده‌آل‌ترین مقدار برای فاصله مجدد در این بار کاری است.

در این نمودار، مانند سایر نمودارها، واریانس فاصله استفاده مجدد زیاد است.

با توجه به نمودار بدست آمده، می‌توان عبارات زیر را نتیجه‌گیری کرد:

- کمترین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر صفر است که حاکی از آن است که دو دسترسی پشت سر هم به یک خانه از حافظه اتفاق افتاده است.
- بیشترین تعداد آدرس متمایز بین دو دسترسی متوالی به یک خانه از حافظه، برابر ۱۶۵۶۰۷۵ است.
- میانگین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۳۲۴۰۰۷ می‌باشد. این مقدار نسبت به میانگین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه سایر بارکاری‌ها به طور قابل توجهی بیشتر است. این بدان معناست که در این بار کاری خاص، محلیت زمانی و مکانی در دنباله دسترسی به حافظه نسبت به سایر بار کاری‌ها کمتر است و این امر موجب می‌شود که در یک بازه زمانی یکسان، تعداد دسترسی‌ها به یک خانه از حافظه نسبت به سایر بار کاری‌ها کمتر باشد.
- واریانس تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۵۷۸۳۲۷۴۰۸۰۸ می‌باشد. با توجه به تراکم بسیاز زیاد داخل یک بازه خاص و واریانس به نسبت بالا می‌توان نتیجه گرفت که تعداد دسترسی به برخی داده‌ها بسیار زیاد و به برخی دیگر از داده‌ها بسیار کم است.
- میانه تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۴۰۹۷۲۲ می‌باشد. این مقدار با توجه با نمودار قابل حدس است زیرا تراکم تعداد داده‌ها در بازه فاصله استفاده مجدد ۵۱۲ هزار بیشتر است.

فایل ۶۶۹A



شکل ۴: فایل ردیابی علی‌بابا ۶۶۹A

مطابق نمودار، پرتکرارترین فاصله استفاده مجدد، فاصله استفاده مجدد ۴هزار است. این امر می‌تواند به دلیل دو دسترسی پشت سر هم مانند یک نوشتن پس از خواندن یا یک خواندن پس از نوشتن باشد.

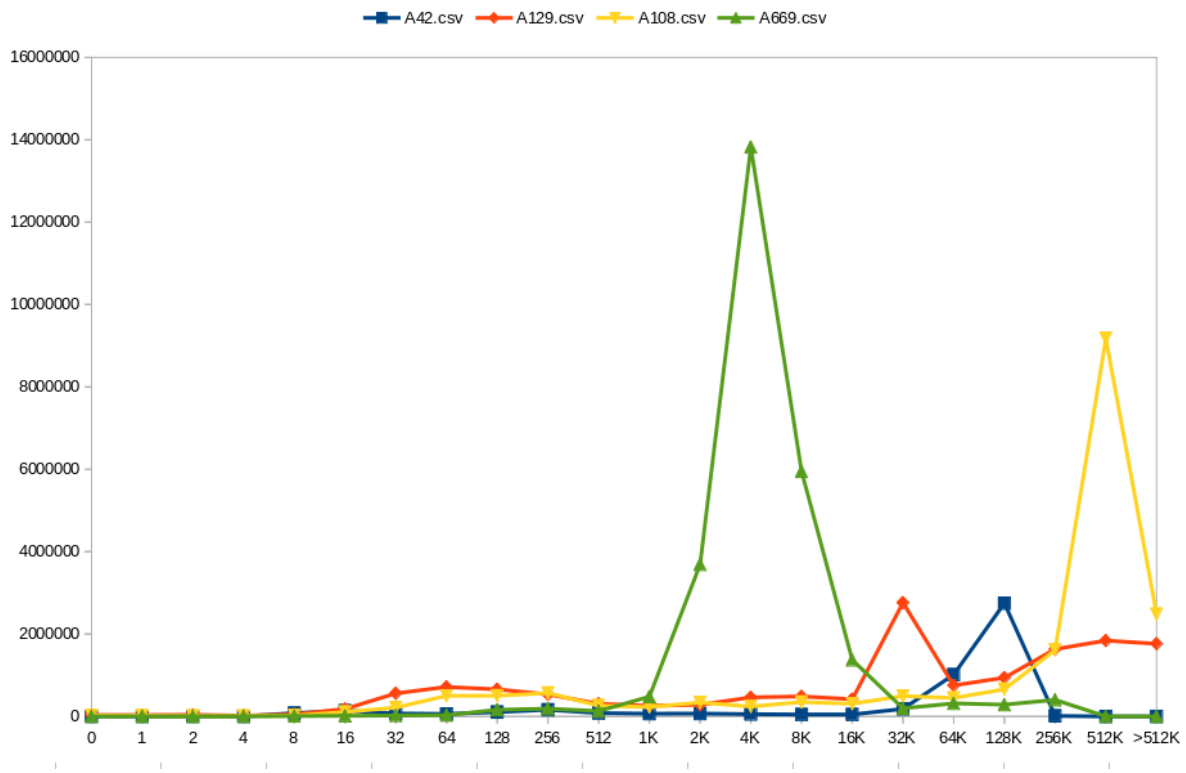
کم‌تکرارترین فاصله استفاده مجدد، فاصله استفاده مجدد در باره ۰ تا ۵۱۲ می‌باشد که به مقدار ناچیزی است و به معنای این است که بین دو دسترسی متوالی به یک خانه از حافظه، احتمال این که کم تر از ۵۱۲ آدرس متمایز وجود داشته باشد بسیار کم است.

در بازه فاصله استفاده مجدد ۰ تا ۵۱۲، همانطور که گفته شد تغییرات ناچیز است است. در حالی که در بازه فاصله استفاده مجدد هزار تا ۴هزار این روند صعودی بوده و رشد شدید و ناگهانی دارد و در ۴ هزار به اوج خود می رسد درحالی که پس از آن یک روند نزولی را طی می کند و در بازه فاصله استفاده مجدد ۸ هزار تا انتها نزولی می‌شود و مجددا در بازه های انتهایی تر این تعداد به صفر میل می کند. با این تفاسیر، می‌توان نتیجه گرفت که بازه فاصله استفاده مجدد ۴ هزار مقدار مناسبی برای فاصله استفاده مجدد در این بار کاری است.
بر خلاف سایر فایل ها در این فایل واریانس داده ها به نسبت کم تر است.

با توجه به نمودار فوق، نتایج زیر قابل استنباط است:

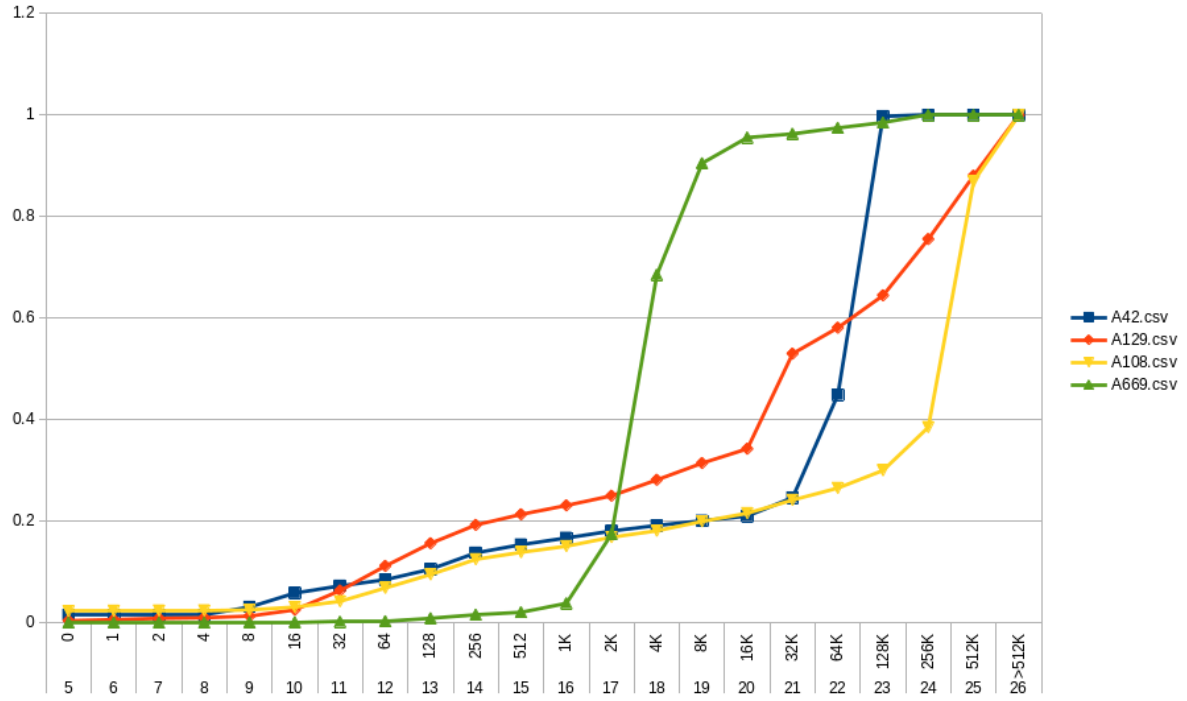
- کمترین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر با صفر است که نشان می‌دهد دو دسترسی متوالی بدون آدرس متمایزی بین آن دو رخ داده است.
- بیشترین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۱۷۸۲۰۹ می‌باشد.
- میانگین تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه، برابر ۷۲۹۲ است. این مقدار از میانگین در سایر بار کاری‌ها به طور قابل توجهی کمتر است. می‌توان نتیجه گرفت که محلیت زمانی و مکانی در دنباله دسترسی به حافظه این بار کاری به مقدار حداکثر خود در بین سایر بار کاری‌ها می‌رسد. همین امر موجب می‌شود که در یک بازه زمانی یکسان، دسترسی‌های بیشتری به یک خانه از حافظه نسبت به سایر بار کاری‌ها انجام شود.
- واریانس تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۳۹۱۲۱۹۵۰۵ می‌باشد. واریانس در این بار کاری کمترین واریانس در بین بار کاری‌های ذکر شده می‌باشد. یک واریانس کم نشان می‌دهد که فاصله‌های استفاده مجدد نسبتاً یکنواخت هستند و الگوهای دسترسی به حافظه منظم‌تر هستند.
- میانه تعداد آدرس‌های متمایز بین دو دسترسی متوالی به یک خانه از حافظه برابر ۳۱۵۳ است. علت این موضوع، تراکم بیشتر تعداد داده‌ها در بازه فاصله استفاده مجدد در بازه ۴ هزار است.

۴ تحلیل تاثیر مقدار میانگین فاصله استفاده مجدد بر دو مؤلفه محلیت زمان و فضایی



شکل ۵: نمودار توأم چهار فایل ردیابی علی‌بابا

با توجه به این عکس برای تمامی فایل‌ها بیشترین میانگین استفاده مجدد در هر فایل در در بازه ای است که بیش تر این تراکم را دارد یا به صورت دیگر می توان گفت اغلب فواصل مجدد در یک قسمت انباشتشه شده اند. حال می‌توان اینطور برداشت که ما پس از طول نقطه اوج هر فایل میان میانگین‌های استفاده مجدد کاهش زیادی شکل گرفته و زیاد کردن طول برای محلیت زمانی و مکانی نیاز نیست، پس می‌توان اینطور برداشت کرد که باید طول پنجره در محلیت زمانی باید ضریبی باشد که به نسبت داده های در این فایل ها و همچنین نقاط بیشینه نسبی آنها وابسته است و به طور کلی بر حسب xt بیان شود که x نقطه بهینه برای میانگین فاصله بین هر درخواست با درخواست دیگر در مجموعه بارکاری های ما بوده و t مقدار متوسط زمانی که بین دو درخواست پشت سر هم انجام می شود.



شکل ۶: نمودار چگالی احتمال توأم چهار فایل ردیابی علی‌بابا

در این تصویر نیز که چگالی احتمال‌ها را محاسبه نموده می‌توان دید که نقطه‌ای که فواصل را در بین این بارکاری‌ها بیشینه می‌کند نقطه ای است که پس از آن شیب چگالی احتمال آن رشد چندانی نداشته است و انتخاب این نقطه به طور میانگین میان بارکاری ها بهترین نقطه برای بهینه سازی cache است، بدین شکل که پس از آن پیشرفت چشمگیری در مجموع این استفاده‌های مجدد نداریم که به صرفه نیست تا این سربار را متحمل شویم. همانطور که در قسمت قبلی نیز به آن رسیدیم و برای محلیت مکانی نیز می‌توان اینگونه تحلیل کرد بهترین اندازه‌ای که برای حافظه نهان می‌توانیم در نظر بگیریم که به بهینه‌ترین حالت برسیم xs است که s همان اندازه‌ای است که هر درخواست در حافظه اشغال می‌کند در این بارکاری‌هایی که در حال بررسی آن‌ها هستیم و x همان نقطه ای است که به طور میانگین نقطه بهینه بین تمام بارکاری های داده شده است که با توجه به تعداد داده های هر فایل و نمودار چگالی احتمالی آن ها بازه های ۶۴ هزار و ۱۲۸ هزار بهترین انتخاب هستند، در نتیجه بهتر است اندازه حافظه را با این مقدار ۱۲۸ هزار برابر s گرفته تا به بهترین حالت برسیم.