دانشگاه جامع علمی –کاربردی
مرکز آموزش عالی جهاد دانشگاهی مشهد

پارتیشن بندی زمان اجرا برای حلقه های موازی برای کاهش وابستگی ترافیک cache

زیر نظر استاد محترم :

مهندس مهدوی

مترجم : عباس فدایی

تابستان 85


بسم الله الرحمن الرحيم

چکیده:

برای بدست آوردن پهنای باند پروسسور به حافظه و تاخیر احتیاجات multiprocessor ها بطور فزاینده ای سلسله مراتب حافظه را ترکیب میکند. تمهیدات پارتیشن بندی زمان اجرا ضروری است برای کاهش هزینه های ارتباط و همزمانی و بدست آوردن کارایی قابل قبول . پارتیشن بندی داده های سازگار پیشنهاد شده است برای کاهش زمان اجرای برنامه های موازی بوسیله کاهش اتصالات بین پروسور برای حلقه های موازی تکراری . ADP میتواند مجتمع شود با یک پروسسور کاهش دهنده ارتباط برای برنامه های موازی کننده جاری. یک مدل چند برنامه ای تعریف شده است برای تحلیل سازنده برنامه هایی که منجر شده است به ارتباط بین پردازنده ای .

یک مدل برای حلقه های موازی تکراری تعریف شده است برای محدود کردن الگو های ارتباطی داخل یک برنامه . یک علامت گذاری انتخاب شده است برای محدود کردن ارتباط یک مجموعه داده های سراسری . پارامترهای ارتباط محاسبه شده اند بوسیله آزمودن ایندکس های دسترسی به آرایه و تنظیم شده برای منعکس کردن معماری سیستم بوسیله جبران سایز خط cache . این ارزشها استفتده شده اند برای تولید پارتیشن های مستطیلی و شش گوشه ای که ارتباط بین پروسی را کاهش میدهند. آزمایش های اندازه گیری پیچیدگی ارتباط روی کارایی برنامه بوسیله استفاده از سیستم ADAPT اجرا میشوند. یک برنامه پارتیشن بندی که بطور اتوماتیک کدهای Fortran را بازسازی میکند . ADP نشان داده شده است بعنوان مافوق استراتژی پارتیشن بندی موجود است بوسیله اندازه گیری زمان اجرا روی Multimax 16 بیتی. نتایج شبیه سازی روی یک شبیه ساز چند پردازنده ای نشان میدهد که ADP متد های پارتیشن بندی جدیدی داده بنا نهاده است بوسیله کاهش ترفیک cache که بموجب آن کاهش پیدا میکند نسبت خطا به بیش از 30 درصد.

1- معرفی

در شروع پردازش موازی : تحقیق در کامپایل برای ماشین های موازی متمرکز شده است روی تعریف و استفاده از همسویی. آزمون روی یک سیستم چند پردازنده ای واقعی آشکار کرد مشکلات جدیدی را که آدرس دهی شده بودند برای تسهیل استفاده وسیع از سیستمهای چند پردازنده ای . بطور خاص ,ارتباط بین پردازنده ای و هماهنگی میتواند تنزل دهد بدست آوردن سطح کارایی را وقتی که موازی سازی کافی ای در دسترس است.

حافظه مشترک پردازنده ها عرضه میکند یک ودل آشنا برای برنامه نویسان . هرچند توافقهایی لازم است برای ساختن یک چند پردازنده ای قابل توسعه. سیستمهای چند پردازنده احتیاج دارند به سلسله مراتب پیچیده ای برای اشتراک پهنای باند حافظه به پردازنده و احتیاجات راکد. برنامه های موازی باید پارتیشن بندی شده باشند برای بهره برداری از پیکربندی سلسله مراتب حافظه مربوط به سیستم . کاربعدی برای سیستمهای چند پردازنده درگیر شدن با بازسلزی خودکار برای به حداقل رساندن اتصالات و هماهنگی هاست.

در پارتیشن بندی برنامه های موازی یک برنامه شکافته میشود به task هایی که میتوانند با هم اجرا شوند روی یک پردازنده . پارتیشن بندی برنامه های ایستا جذاب شده از موقعی که پارتیشن بندی برنامه مشخص شده در فاز کامپایل به موجب آن حذف یک منبع از زمان اجرای بالاسری .وانگهی پارتیشن بندی ایستا کاهش میدهد اتصالات بالا سری مرتبط با الگوهای دینامیک را که کوشش میکنند استفاده از پروسسور را بهبود ببخشند . پارتیشن بندی داده شازگار یک روش پارتیشن بندی ایستا است که بطور بهینه ای تقسیم میکند یک مجموعه داده بزرگ را از میان پروسسور های اجرایی هماهنگ برای کاهش اتصالات احتیاجات برای ساختن برنامه ای موازی ویژه ای کامپایلر هایی هماهنگ ساز برای بر اساس این دوقضیه هستند . اولا هماهنگ سازی قابل بهره برداری که مقیاس شده اند با سایر شکل که در چرخه های DO بطور گسترده ارائه شده اند و بنابراین توسعه تکنیکهای موازی سازی مناسب برای حلقه های DO اولین اهمیت هستند.
دوم با وجود آن تکنیکهای موازی سازی براساس قالبهای کاری تئوریک تحلیل وابستگی هستند . یک رشته از تکنیکها برای ساختارهای برنامه نویسی رخدادهایی که میخواهند توسعه پیدا کنند. کار ما روی کاهش اتصالات بین پردازنده ها همچنین بر پایه این دو فرض هستند. از این گذشته از وقتی حلقه های DO بیشترین تمایل برای تحلیل کردن شده اند شروع تلاش در به حداقل رسانی اتصالات باید بر اساس حلقه های DO باشند. ADP مفید خواهد بود در توسعه برنامه های موازی قابل حمل . بطور نمونه , کاربران اجرا میکنند برنامه هایشان را روی یک رشته از چند پردازنده ها . بیشتر اوقات آنها مجبور میشوند برای تغییر برنامه هایشان بطور وسیع برای بهره برداری از سلسله مراتب حافظه خاصی . یک کامپایلر ایده آل برای برنامه های موازی قابل حمل باید خذف کند این بار را برای کاربر . ADP که پارتیشن بندی میکند ساخت برنامه را بطور اتوماتیک برای چند پردازنده داده شده. همچنین کامپایلر .

شکل 1 را ببینید که یک نمونه از یک چرخه موازی تکراری است . که شامل حلقه های DO ترتیبی بیرونی است و چرخه های موازی داخلی است که update هایی انجام میدهند روی یک

آرایه بزرگ داده . چرخه DO ترتیبی خارجی مجبور میشود مقارن سازی شود در پایان هر تکـرار خارجی و تحمیل ترتیب update ها را به عناصر خاصی از آرایه را . تکرار حلقه مـوازی داخلـی ممکن است اجرا شود در هر درخواست . یک مثال از اینچنین حلقه ای در برنامه های عـددی حـل معادلات تفاضلی جزئی در طول استراحت است . مجموعه شروع های آرایه که برمیگردد به عنـوان الگو برداری . پارتیشن بندی داده در حلقه های موازی تکراری تقسیم میکند مجموعه های ایندکس حلقه های داخلی را در بین پردازنده ها . هر پردازنده اجرا میکند همان کد را روی یـک منطقـه از مجموعه داده و بطور نمونه ارتباط فقط با پردازنده هایی که روی مناطق همسایه کار میکنند.

در پروسیجر پارتیشن بندی خودمان مجموعه استنسل ها(که از برنامه گرفته شده اند) و اندازه خط cache (یک پارامتر معماری) استفاده میشوند برای محاسبه پارامترهای اتصالات . پارتیشن بندی مستطیلی یا شش گوشه ای بدست آورده میشوند از ارزش پارامترهای اتصالات . در ساخت کار قبلی مستطیلها و شش گوشه های نامنظم نشان داده شده اند بعنوان برتر از مربع ها و شـش گوشـه هـای منظم اگر ارتباط مساوی نباشد در جهت های مختلف یا اگر سایر خط cache چند برابر از انـدازه هر یک از چند نقطه داده باشد . در مورد ارتباط نابرابر و اثر خط cache , الگـوریتم مـا ممکـن است ترکیب شود با دو کاندید برای حصول یک پارتیشن بهینه که یک مربع باشد . ما نشان میدهید که ارتباط میتواند محدود شود یا آزمایش زیر نویس آرایه .

کا پارتیشن داده قبلی و آنچه مربوط به اوست به ADP ارائه شد سپس فرضیات در مورد سیـستم و برنامه مطرح شد . یک روش تحلیل حلقه های موازی تکراری برای ساخت الگوهای ارتباط توصیف شد. برای بدست آوردن اجازه برای مقایسه ها بین پارتیشن ها یک سیستم متریک ارتبـاط تعریـف شد . الگوهای ارتباطی داده شده برای یک سگمنت کد , فرمولهایی هستند که تولید میکنند انـدازه پارتیشنهایی که این انـدازه را کـاهش میدهنـد یـک سیـستم پارتیـشن بنـدی برنامـه ADAPT (automatic data allocation and partitioning tools) بـرای پیـاده سـازی ADP استفاده میشود یک تحلیل برنامه نمونه نشان داده شده است که شامل شبیه سازی نتایج و زمانهای کد از یک Multimax است.

II . عملیت مرتبط

ما سه منبع اصلی از کار قبلی را رسم کردیم . تکنیکهای کامپایلر برای پارتیشن بندی حلقه های موازی به مجموعه های مستقل قابل اجرا روشهایی ایستا برای پارتیشن بندی فضا های تکرار و تکنیکهای پارتیشن بندی برای الگوهای موازی . اولین مجموعه کاغذ ها حلقه ها را می آزماید برای تعیین مجموعه های تکرار که میتواند اجرا شود بدون هیچ تغییر اطلاعات بین مجموعه ها . کا پارتیشن بندی فضای تکرار بطور همزمان رسیدگی میکند موازی بودن و پارتیشن بندی را . کار الگوریتمیک پارتیشن بندی میکند یک ماتریس سراسری را بوسیله کاشیکاری آن با تعدادی از شکلهای هندسی استاندارد مثل مربع , شش گوشه منظم یا مثلث .

کار کامپایلر های جدید گزارش شد در [4] - [1] . مبدا های آرایه در دستور ضمیمه شد بوسیله مجموعه ای از حلقه هایی که آزموده شدند . مجموعه ایندکی پارتیشن بندی شد به مجموعه های مستقل قابل اجرا که هرکدام از آنها میتواند اجرا شود روی یک پردازنده مجزا بدون ارتباط بین پردازنده ای . در مقابل از کار قبلی , که تولید میکند یک متغیی عددی از مجموعه های مستقل اجرایی , پارتیشنهای ADP یک حلقه موازی برای بدست آوردن استفاده از همه پروسسورها , با ارتباط کوچک (و نه صفر) ارتباط بین پردازنده ای است.

ADP مکمل پارتیشن بندی مجموعه های مستقل اجرایی است. وقتی یک تعداد نا کافی از مجموعه های مستقل اجرایی قابل دسترس هستند , ADP استفاده شده روی مجموعه های اجرایی مستقل منحصر به فرد , افزایش میدهد استفاده از پروسسورها را روی هزینه چند ارتباط .

یک مقدار کار خوب اجرا شده در کاشیکاری فضاهای حلقه های ترتیبی تکرار[5]- [9] . یک thread معمولی در همه این اوراق ملاحظات همزمان موازی سازی و پارتیشن بندی با استفاده از موضوع مستقل محور است. بر خلاف روشهای قبلی پارتیشن بندی داده ها ی سازگار یک بهینگی برای ساختار موازی است . پارتیشن بندی تولید شده بوسیله ADP بهینه است تحت سیستم متریک . کار قبلی کوشش نمی کرد برای مقایسه کیفیت پارتیشن بندی یا اثبات نا بهینگی . کشف موازی سازی یک فعالیت front – end است , i.e. و امکان پذیرش است بدون ملاحظه ویژگی چند پردازنده ای هدف . پارتیشن بندی موثر یک فعالیت back- end است که به دانش اجرایی ماشین احتیاج دارد . بنابراین در باره دو فعالیت صحبت میکنیم .

ترافیک Bus در سیستمهای چند پردازنده ای مربوط به دو مجموعه است :

خطاهای هر یک از پردازنده ها و ترافیک هماهنگی cache [10]. هر یک از دو Gallivant , at-al. [10]و [12] . کارمان به اشتراک میگزارد یک هدف عمومی برای بـه حـداکثر رسـاندن موفقیت نسبت در جابجایی برنامه هاست. آنها تعریف میکنند یک پنجره مرجع برای یک وابستگی از متغیر هایی که ارجاع داده شده اند بوسیله منبع و حفره وابستگی . بعد از اجـرای منبـع وابـستگی ذخیره کردند پنجره ارجاع داده شده مشترک در cache تا بعد از اجرای حفره اجـرا میـشود کـه بهبود میبخشد تعداد نسبت را . این بهینه سازی کوشش میکند به حد اکثر رسـاندن دوبـاره اسـتفاده کردن از داده های cache را در متن منتهی به i.e.. cache . با کاهش خطاهای تک پردازنـده ها .ADP انجام میشود بدون ملاحظه تاثیرات اندازه cache و به حداقل رساندن تعداد متغیرهای به اشترتک گذاشته شده در پردازنده ها و بدین ترتیب کاهش ترافیک هماهنگی cache .

کارهای وابسته روی پارتیشن بندی در سطح الگوریتم [16]-[12] بحث شـده اسـت . [16] ارائـه میکند یک رفتار کامل از موضوع را , شامل فرمان از تحلیل سه گانه الگو/پارتیشن/ معماری بـرای بدست آوردن اتحاد اثرات چندین فاکتور روی یک کارایی الگوریتم بدون یک مدل. کارهای مهم بعدی شامل آن است gallivant . [12] کسی که کوشـش کـرد بـرای متعـادل سـازی اثـرات locality و همزمانی و برداری کردن روی کارایی الگوریتمهای جبـر خطـی انبـوه . شـکل دادن الگوریتمهای بلوکی که بهبود میبخشند کارایی را برای locality های افزایش داده شده.

کارهای قبلی روی الگوریتمهای پارتیشن بندی میسر کردند راهنمایی هـای عـالی بـرای صـراحی کردن الگوریتم موازی و توسعه برنا موازی . ADP یک ترجمه از این تکنیکهای الگوریتمی است به تکنیکهای کامپایلی که بطور اتوماتیک تولید میکند پارتیشنهای برنامه ای بهینه . بوسـیله حـذف هردو با زمان کوتاه توسعه برنامه های موازی و بهبود کارایی برنامه های موازی .

این اوراق توسعه میدهند الگوریتمهایی که مناسب کامپایلرهایی که کشف میکنند الگوهای ارتباطی را در سگمنت کد و تنظیم میکنند شکل پارتیشنها را به کاهش ارتباط از این سگمنتها . این کار که در اینجا ارائه شد ایده هایی دارد از موضوعات قبلی به پارتیشن بندی داده . ما از شـکلهای هندسـی استفاده میکنیم برای کاشیکاری ماتریس سراسری , شبیه الگوریتم پارتیشن بندی .هر چنـدما تغیـر می دهیم شکل پارتیشن ها را برای جبران ناهمگونی در الگوهای ارتباطی کـه تولیـد کننـده قـبلا رسیدگی کرده. نشان خواهد دادچند نتیجه که ارئه شد در[16]و[14] موارد خاصی از ADPهستند.

III- مفروضات مدل

A: مدل چند پردازنده

استراتژی پارتیشن بندی که در اینجا ارائه شده موثر است برای یک رشته از سیستمهای موازی شامل سیستمهای حافظه مشترک و حافظه توزیع شده . استثنا های قابل ذکر مالتی پروسسورهای با حافظه مشترک بدون cache یا با cache مشترک است .e.g Alliant FX-8. برای محدود کردن و تشریح بهبود دستیافته در ارتباط پردازنده بوسیله استفاده از پارتیشن بندی داده ها ی سازگار , ما متمرکز خواهیم شد روی چند پردازنده های مشترک با copy-back حفاظت شده cache . اینچنین چند پردازنده هایی شاید از نظر اقتصادی موفق باشند , .e.g , تقارن تابعی و Multimax .

سیستمهای چند پردازنده ای مبتنی بر Bus بطور فزاینده ای استفاده میکنند از پروتکل copy-back در هر مانیتور cache , یا جاسوسهای روی bas سیستم برای اطمینان از هماهنگی حافظه پروتکلهای تجسس copy-back ترافیک را کاهش میدهند روی Bas از وقتی update های تکراری مکان مشترک ممکن بوسیله یک پروسسور موجب هیچ تراکنش Bas نمی شود . بزرگی دسترسی هیچ کدام از دیگر پردازنده ها در آن منطقه .

فرض بطور کافی cache حفاظت شده بزرگ , ترافیک همزمانی cache کامپوننت غالب در ترافیک Bas است. [17] , بنابراین ترافیک Bas میتواند فرض شود به سازگاری بطور سراسری در اتصالات نتایج جزئی بین پردازنده های عملیاتی بطور هماهنگ هدف الگوهای پارتیشن بندی ما به حداقل رساندن موضوع ترافیک هماهنگی cache به محدودیت اختصاص . task های با اندازه برابر به هر پردازنده برای تعادل بارگزاری است .

B – مدل برنامه

این اوراق بر حلقه های موازی تکراری متمرکز شده است . این حلقه ها تعریف شده اند بعنوان دو محل برای حلقه ها و یک بدنه کد.

محل بیرونی هماهنگ است با حلقه های ترتیبی . محل داخلی یک جفت از حلقه های موازی است هر پیمایش مجموعه تکرار . برای حلقه های داخلی فضای تکرار تعریف شده است بعنوان $I = I_1 * I_2$. هر تکرار حلقه ترتیبی بیرونی یک چرخه است . بدنه کد میتواند هر مجموعه از دستورالعمل فابل اجرای B باشد , بنابراین طول بعنوان یک نگاشت از مجموعه داده از برنامه به فضای تکرار داده میشود . فقط محدودیتهایی که روی کد برنامه قرار گرفته اند آن محاسبات و الگوی دستیابی هستند که یکسانند در هر نقطه در I .

برای این کاغذ , بدنه کد محدود شده است در بروز رسانی یک نقطه داده A , (i , j) با یک ماتریس سراسری یک یا دو بعدی A . هر بدنه کد ممکن است استفاده شود , اما این مثال بطور گسترده ای داده شده است نگاشت مجموعه داده به فضای تکرار . برای هریک ارجاع حافظه به ماتریس سراسری به داخل سمت راست update بدنه کد , ارجاع از A (i+a,j+b) خواهد بود , جایی که a,b نمونه های عددی هستند. زوج q=(a,b) بردار دسترسی شناخته میشوند . مجموعه همه بردارهای دسترسی برای بدنه مجموعه الگو شناخته میشوند .

مثال حلقه های موازی تکرار که در شکل 1 ارائه شد را ملاحظه کنید . مجموعه تکرار برای هر کدام از حلقه های داخلی است [1..n] . فضای تکرار برای مکان حلقه داخلی است [1..n]*[1...n]. بدنه کد دستوری است که A(i,j) را update میکند .برای این بدنه کد , مجموعه پردازنده های دسترسی برای سمت راست دسترسی ها به آرایه {(2,0),(1,0),(-1,0),(-2,0),(0,1),(0,-1)} مجموعه الگوست , هرکامپوننت از مجموعه الگو یک بردار دسترسی است . این ساختار برنامه انتخاب شده است به خاطر این فاکتورها :

- این سازه ها پدیدار شده اند در برنامه های کاربردی عددی و .e.g و راه حلهایی ناهمگون به معادلات جزئی مختلف [13] , مدلهای مستمر[18] و هموارسازی شکلها [19] .
- بروز رسانی هر نقطه در حلقه داخلی یک عمل مستقل است , بنابراین موازی سازی در مسئله گسترده است از موازی سازی ذاتی در موازی سازی متصل چند پردازنده.
- حلقه DO خارجی احتیاج دارد که ارزشهای استفاده شده در بروزرسانی هر نقطه داده آن محاسبه شده در چرخه قبل باشد .

یک قسمت مجتمع شده در این کار توانایی بهینه سازی در ارائه اتصالات نامتقارن است . اثرات مستقل از ماشین مثل اثرات اندازه خطای cache , میتواند تحمیل کند اتصالات ناهمگون روی یک الگوی متقارن را . اگر چه الگو های متقارن اتفاق می افتد در بعضی کدها , برای مثال در بعضی از کدها هموارسازی image ,[19] , الگوی

S={(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2),(3,1)}

استفاده میشوند برای محاسبه x وy مشتق جزئی از حرکات حاشیه ای . مطابق آنچه خواهیم دید یک شش گوشه نامنظم کاهش میدهد ارتباط را برای این الگوی جزئی .

مطابق تعریف [20] دو نوع از ارتباطات بین پردازنده ای تعریف شده است , ارتباط داده و ارتباط کنترل . ارتباط داده هنگامی بوجود می آید که پردازنده داده که برای محاسبه نیاز دارد تعویض کند . ارتباط کنترل وقتی بوجود می آید که پردازنده ها اطلاعات کنترل را به task های انجام داده شده تعویض میکنند (مثل همگام سازی در یک بروزرسانی مانع یا سمافور)

از وقتی C_p , داده زمان ارتباط , مطابق شده بااندازه مسئله , C_p محاسبه شد برای توده ای از جریمه های ارتباطی پرداخت شده برای مشکلات بزرگ . در یک پارتیشن بندی ایستا, C , زمان ارتباط کنترل , یک تابع از تعدادی از پردازنده هاست و مستقل از پارتیشن است . بنابراین پارتیشنها محاسبه میشوند منحصرا روی زمان اتصال داده شان.

IV – اتصال محدودیت

یک روش برای الگوهای اتصال محدودیت داخل یک برنامه لازم است برای ساختن پارتیشن داده ای با حداقل اتصالات این قسمت ارائه میکند ساختهایی روی شکلهای پارتیشن و تعریف میکند آینده مجموعه الگو که میتواند استفاده شود در الگوهای اتصال محدود.

A – پارتیشن

پارتیشنهای معتبر مورد نیازند برای جبران شرایط مقابل مثل [9] . پارتیشن بندیهای عملی مثل مستطیل هل و مربع ها مشخصات زیر را جبران میکنند .

1 - موزائیک کاری : پارتیشن باید کاشیکاری کند فضای تکرار را بطوری که هر نقطه روی فضای تکرار متناظر شود با یک پردازنده. n^2 داده شده , اندازه فضای تکرار و P تعداد پردازنده هاست , s تعداد تقریبی داده ها برای هر قسمت است . $s = \frac{n^2}{p}$

2– معین بوسیله پیرامون : برای مطمئن شدن از کاشیکاری و تولید کد ساده , محیط هر قسمت با یک خط ترکیب میشود $\{l_i\}$, که اتلاق میشود (نه خطوط عمودی) شماره های مستدل هستند . احتیاجات گذشته مطمئن میکردند که سگمنت خطوط از نقاط داخل فضای تکرار مجزا نمی گذرند.

3– تقارن : قسمت خطوط $\{l_i\}$ میتواند جفت شود بنابراین آن قسمتها در هر جفت جهت یابیهایی دارند $\{l_i, l_j\}$ مثل $|l_i| = |l_j|$ و l_i موازی l_j است .

B – تولید وزنهای ارتباط

1 - مجموعه اهداف و سایه ها :یک کمان دسترسی برای یک بردار دسترسی q=(a,b) یک بردار از یک آرایه (i,j) به(i+a,j+b) است . مجموعه هدف مربوط به یک نقطه داده y تحت یک الگوی داده شده همه نقاط داده لازم هستند برای محاسبه y .

تعریف l , $y = (i, j) \in A$, ماتریس سراسری $s = \{(a_0, b_0),...,(a_k, b_k))\}$ مجموعه الگو هستند . سپس مجموعه هدف به y تحت s است. $T_s(y) = \{(i + a_0, j + b_0),...,(i + a_k, j + b_k)\}$

سایه قسمت p تحت یک الگوی s مجموعه نقاط خارجی p که مورد لزو م برای محاسبه نقاطp و اندازه آنها مساوی تعداد نقاطی که از پردازنده های دیگر آورده شده اند برای بروز رسانی نقاط داده در p است . برای یک قسمت داده شده p و مجموعه الگوی s سایه p تحت s است

$$p_s = \{x \mid x \notin p, \exists y \in p_{s.t} .. x \in T_s(y)\}$$

2- یک برادار دسترسی تکی : با ملاحظه ارتباط ایجاد شده بوسیله یک جفت قسمت خط $\{l_i, l_j\}$ با یک طول L در طی یک بردار دسترسی از q بزرگ متمایل به زاویه q با نسبت $\{l_i, l_j\}$. اتصال ایجاد شده بوسیله $\{l_i, l_j\}$تعریف شده به تعداد دسترسیهای گذشته از l_i یا l_j از داخل P است . وزن ارتباط مربوط به قسمت خط $\{l_i, l_j\}$اتصال ایجاد شده در واحد طول برابر است با افزایش در نتایج ارتباط از یک واحد افزایش در هر دو l_i و l_j است .

وزن اتصال در واحد طول تعداد دسترسیها در چرخه خارجی از قسمت داده شده در طول یک واحد طول قسمت جفت خط.

برای مثال , ملاحظه کنید که کمان دسترسی که مجتمع شده است با بردار دسترسی (3,2) (شکل 3) تجزیه $q \sin q$ در طول محور افقی میدهد مقدار اتصال اضافی محدود شده برای همه نقاط داده در طول حاشیه عمودی پارتیشن که دو تاست . تجزیه در طول محور عمودی میدهد مقدار اتصال اضافی برای هر داده در طول حاشیه افقی .

یک بردار دسترسی داده شده و یک پارتیشن قراردادی احاطه شده است مربوط به زوجهای m از قسمت خط مربوط به طول l_i , $1 \le i \le m$, نظریه 2 میدهد اتصالات ایجاد شده در طول چرخه بوسیله هر قسمت .

نظریه 2 : فرض کنید یک بردار دسترسی به اندازه q و یک پارتیشن اختیاری احاطه کننده از m جفت شده است از قسمت خط با هر طول l_i , $1 \le i \le m$ تعریف میکند $n_i q \sin q$برای همه $1 \le i \le m$و اتصال c_p دلخواه روی هر حوزه یک تعریف است . $c_p = \sum_{i=1}^{m} l_i n_i$

اثبات : هر زوج از قسمت خط حساب میکند برای اتصال $n_i l_i$ بوسیله قضیه 1 . بنابراین همه اتصالات برای همه زوج خطهای m مصابق بالا . اگرچه بعضی از نقاط داده نزدیک به گوشه های قسمت ممکن است محاسبه شوند دوباره در این تحلیل . بنابراین حدود طبیعی از نتایج اگرچه دوره نمایش دادن نقاط گوشه فقط تابعی از بردار دسترسی هستند و مستقل از اندازه قسمت .

ملاحظه کنید یک الگوی پارتیشن بندی مستطیلی آنجایی که هر مستطیل اندازه گرفته میشود v و h در طی محورهای عمودی و افقی . مطابق آنچه در شکل 4 نمایش داده شده است , وزنهای اتصال مجتمع شده تعریف شده است $n_v = n_1$ و $n_h = n_2$ اولین و دومین عناصر از بردار دسترسی هستند .

حدود اتصال معین میشود بوسیله نظریه 2 $c_p = hn_h + vn_v$

برای شش گوشه ها , یک مثال محاسبه c_p ممکن است در دوره های با گرایشهای افقی , قطری معکوس و قطری . دو جفت مجرای قسمت خط موجود است , که طولشان b و h و d برای افقی , قطری و قطری معکوس . معادله c_p با n_h و n_b و n_d تعریف شده اند مانند n_v و n_h .

$$c_p = hn_h + bn_b + dn_d$$

3 – بردارهای دسترسی چند گانه . یک روش برای تعیین وزن اتصالات برای یک دسترسی تک , ما اکنون برگیگردیم به یک مجموعه از بردارهای دسترسی k $\{q_1, q_2, ..., q_k\}$.

با روش ارائه شده در قسمت قبل ما میتوانیم تولید کنیم $N_k = \{n_{h1}, n_{h2}, ..., n_{hk}\}$ و $N_v = \{n_{v1}, n_{v2}, ..., n_{vk}\}$ دوروش ارائه شده برای تولید n_h و n_v ترکیب کردن وزن اتصالات از N_h و N_v ملاحضه اصلی برای صحت ارجاع همان جفت نقاط داده است .برای مثال ملاحظه کنید مجموعه الگوی ((1,2),(1,3)) را .برای بعضی ازنقاط A(i,j) . فرض کنید اولین بردار دسترسی احتیا ج دارد که A(i+1,j+3)واکشی شود برای بروز رسانی , فرض کنید A(i,j+1)همچنین در داخل همان قسمت مثل ,A(i,j) دومین بردار دسترسی احتیاج دارد به A(i+1,(j+1)+2)که همان A(i+1,j+3)است بطور صحیح واکشی میشود .

اولین روش عمل میکند همه دسترسی ها را بصورت مجزا و نامیده میشود ساخت افزایشی .

رویه فرض میکند که هر بردار دسترسی شرکت میکند در ترافیک bus . ساخت افزایشی محدوده بالاتری روی ترافیک هماهنگ است .

نظریه k – 3 داده شده بردارهای دسترسی $\{q_1, q_2, ..., q_k\}$ با کامپوننتهای اتصالی افقی $\{n_{h1}, n_{h2}, ..., n_{hk}\}$ و کامپوننتهای اتصالی عمودی $\{n_{v1}, n_{v2}, ..., n_{vk}\}$ وزن اتصال متمایل به استفاده ساخت افزایشی داده شده بوسیله

$$n_h = \sum_{i=1}^{k} |n_{hi}| and n_v = \sum_{i=1}^{k} |n_{vi}|$$

اثبات : برای هر $q_i \in \{q_1, q_2, ..., q_k\}$ نقطه داده $|p_{qi}|$ احتیاج دارد به جابجایی به پردازنده برای یک مستطیل دلخواه با عرض h و طول v

$$|p_{qi}| = h\sum|n_{hi}| + v\sum|n_{vi}| - \sum|n_{hi}||n_{vi}|$$

و از وقتی n_h وزن اتصال ادغام شده با حاشیه افقی $n_h = \sum|n_{hi}|$ مشابه آن $n_v = \sum|v_i|$ و بنابراین ادعا ثابت میشود

در مقایسه روش دوم که max_min construction نامیده میشود بهترین مورد رفتار را دارند . هر نقطه خارجی احتیاج دارد که فقط واکشی شود برای راضی کردن همه ارجاعات داخل چرخه جزئی حلقه . در این مورد فقط بزرگترین وزن ارتباط داده شده شرکت میکند در بارگزاری bas سیستم.

نظریه 4 : k داده شده و بردارهای دسترسی $s=\{q_1,q_2,...,q_k\}$ با اتصالات افقی $\{n_{h1},n_{h2},...,n_{hk}\}$و اتصالات عمودی $\{n_{v1},n_{v2},...,n_{vk}\}$ تعریف شده اند.

$$n_h^+ = \max(\{n_{hi} \mid n_{hi} \geq 0\} \cup \{0\})$$
$$n_v^+ = \max(\{n_{vi} \mid n_{vi} \geq 0\} \cup \{0\})$$
$$n_h^- = \min|\{n_{hi} \mid n_{hi} \geq 0\} \cup \{0\}|$$
$$n_v^- = \min|\{n_{vi} \mid n_{vi} \geq 0\} \cup \{0\}|$$

وزن اتصال خواهد بود $n_h = n_h^+ + n_h^-, n_v = n_v^+ + n_v^-$

اثبات : از تعریفهای p_s قسمتهای $n_h^+, n_v^+, n_h^-, n_v^-$ و ساخت مستطیل EFGH در شکل 5 , $p_s \subset (\text{EFGH/ABCD})$ و $p_s \cap ABCD = f$ بنابراین $p_s \subset (\text{EFGH/ABCD})$ خواهد بود . فورا نتیجه خواهد داد

$$|p_s| \leq |EFGH| - |ABCD|$$
$$= (n_v^- + h + n_v^+)(n_h^- + v + n_h^+) - hv$$
$$= h(n_h^+ + n_h^-) + v(n_v^- + n_v^+) + E$$
$$E = n_h^+ n_v^+ + n_h^+ n_v^- + n_h^- n_v^+ + n_h^- n_v^-$$

بردارهای دسترسی هستند و کامپوننتهای افقی و هستند و بعنوان سایه آن نشان داده میشود . سپس وقتی . این دلالت میکند که

بطور مشابه .

بطور مشابه نظر به اینکه بردارهای دسترسی که کامپوننتهای عمودی و دارند میتوانیم نشان دهیم

خلاصه این چها معادله که

این بدست می آید

چون وزن ارتباط ادغام شده است با حاشیه افقی است , .بطور مشابه

وبنابراین ادعای ما ثابت میشود.

هر دو رویه های سازنده , ارتباط که خطای دوره است . چون در همه موارد خطای ساختن دوره وابسته به اندازه گیریهای مستطیل است, بزرگی در بر نخواهد داشت بهینگی در هیچ رویه . بنابراین از چشم پوشی خواهیم کرد در نتیجه این قسمت

استفاده میشود وقتی که احتمال این که یک نقطه داده در بین دو مربع

از نقاط دارای ارزش باشد زیاد است . این احتمال وابسته به نسبت قسمت مشترک به قسمت غیر مشترک داده است . برای برنامه های ما دانه های task درشت است , بنابراین نسبت قسمت مشترک به غیر مشترک داده کوچک است . به این علت ,

استفاده میشود وقتی نتایج آزمایشی مطرح میشود .

4 – وزنهای ارتباطی شش گوشه ای : تعیین وزنهای ارتباط در واحد طول برای یک پارتیشن شش گوشه ای یک سخت تراز پارتیشنهای مستطیلی است . برای یک بردار دستیابی واحد یک تحلیل مثل مستطیلی انجام میشود برای وزن ارتباط در دوره ای h وb و d در اندازه گیریهای افقی , قطری و قطری معکوس مثل مثال 4 . تنها تفاوت در تجزیه بردارها به دو مجموعه بردار عمودی و افقی است به خوبی فضاهای قطری و قطری معکوس . تجزیه ارجاع داده میشود به

کارهای الگوریتمیک قبلی روی شش گوشه ها استفاده میکردند از شش گوشه های منظم با زوایای 60 درجه بیرونی . در یک طرح یک بخش خطی با زاویه 60 درجه به خطوط داده عمودی گذرانده میشود . برای نگهداری تقارن و مفروش کردن تحمیل میکند , بخشهای خطی ناهموار غیر افقی معرفی میشود در هزینه قابل توجه مجموع محاسباتی زمان اجرا . این قبیل شش گوشه ای ها راضی نمی کنند شرایط خطوط مستقیم محیطی را . در عوض شش گوشه ها پیاده سازی در این قسمت بین قسمت عمودی و افقی نشان خطوط 45 درجه دارند . این شش گوشه ها

کمترزمان اجرا دارند . اگرچه شش گوشه منظم 45 درجه ای فضای بیشتری نسبت به شش گوشه های منظم 60 درجه ای میگیرند و هزینه اتصال زیادی دارند . بردار b و بردار d نسبت 45 درجه به بردار h و بردار v دارند برای استفاده از شش گوشه های منظم 45 درجه ای . برای یک مجموعه الگوی کامل از یک تعداد دلخواه از بردارهای درسترسی , روشهای ساخت گذشته نگه داشته شده اند . همه تجزیه های منحصر به فرد گرفته شده اند .

استفاده میشوند برای تولید وزن اتصال در واحد طول برای یک مرز پارتیشن با گرایشش.

V – جبران اندازه خط کش .

اوزان اتصالات مشتق شده در قسمت قبلی میسنجند تعداد نقاط داده را که تغییر کرده اند در طی یک واحد طول از یک پارتیشن . اگرچه در اکثر سیستمهای چند پردازنده ای , واحد اصلی از بین cache .

نظریه 5 : برای و و L داده شده تعداد خطوط مورد نیاز در واحد طول در طی یک مرز افقی , داده شده بوسیله

وقتی خطوط cache و حاشیه های افقی پارتیشن منظم شدند بوسیله

وقتی خطوط cache متمایل میشوند با در نظر گرفتن حواشی افقی پارتیشن . تعداد خطوط cache مورد نیاز در واحد طول در طی حاشیه عمودی خواهد بود

اثبات : دو مورد تعریف شد و مشتق شد برای هر مورد . در مورد اول حاشیه پارتیشن فرض شد که تطبیق پیدا کرده است با هر خط مرزی (شکل 6) این مورد وقتی پیش می آید که اندازه گیری ستون آرایه و v محدود شده اند به چند برابر اندازه خط cache .آزمودن سایه یک بردار دسترسی در یک حاشیه افقی مطابقت داده شده , اندازه سایه بوسیله(5,1) داده شده است .

در مورد دوم , حواشی خط cache بطورت رندوم قرار گرفته اند با نسبت حاشیه افقی (شکل 7) . این مورد وقتی پیش می آید که اندازه گیری ستون و v بصورت دستی قرار گرفته است . در این مورد ترکیبی از دو دوره است . اولی تعریف میشود برای (5,1) و دومی درست است وقتی از اولی کمتر باشد . ملاحظه کنید به یک دسته L را با ستونهای متوالی , هرکدام با یک اریب منحصر به فرد , s در مورد محدوده خط cache احتیاج دارد بوسیله یک ستون با اریب s هست و مشاهده کنید که

بنابراین میانگین

خطوط cache اضافی مورد نیاز است . بنابراین نتیجه .

اکنون با ملاحظه ارتباط در طی یک حاشیه عمودی (یا هر حاشیه ای در طی بردار که متعامد بر اساس خط cache است), نقاط داده L در طی حاشیه در همان خط cache است , بنابراین در حقیقت فقط یک حرکت مورد نیاز است برای نقاط L . بنابراین (5,3) تشریح میکند ارتباط را .

برای ارزشهای زیاد از , اگرچه اگر خیلی کوچکتر از L (چنانکه معمولا مورد نیاز است) , متفاوت است ..e.g , دلالت میکند

و

B : جبران شش گوشه ها.

وقتی با ملاحظه پارتیشن های شش گوشه ای , , با شکل , حاشیه هـای b و d زاویه 45 درجه میسازد با خط cache و ماکنون ملاحظه میکنیم اثر اینکه خط cache روی حاشیه b مطابقت داده شده است .

نظریه 6 : برای داده شده, میانگین تعداد خطوط cache مورد نیاز در طی حاشیه bبه قرار زی است

Lپهنای باند در دوره نقاط داده

اثبات : با ملاحظه آن , برای هر فاصله در طی بردار افقی , که تعداد نقاط متصل در طی حاشیه b است . از این , میتواند جانشین باشد برای در (5.2). یک فرمول دوباره طرح میکنیم در موارد شبیه معادله هستند (5,4) .

VI – ساختن پارتیشن

نظریه 7 : اجازه دهید s مجموعه الگویی باشد که و را بعنوان اوزان اتصال میسازد. سپس حداقل است برای مربعات وقتی

اثبات : داده شده است برای پارتیـشنهای مـستطیلی بوسـیله (4,3) . تعـدادنقاط داده بـرای هـر پردازنده , s داده شده در (4,1) و برای مستطیلها است hv=s . با اسـتفاده از ایـن بـرای جانـشین کردن برای s است (4.3) و اکنون یک تابع است برای h . سپس همچنان که دیدید بوسـیله b گرفتن مشتق جزئی با نسبت h , و با فرض اینکه و زمان اتصال به حداقل رسیده است بوسیله (6.1) و بنابراین اعدا نشان داده شد .

بنابراین اتصال معلوم در جهت عمودی و افقی و تعداد نقاط داده برای هر پارتیشن , (6.1) میدهـد طول حاشیه برای یک پارتیشن مستطیلی که اتصالات بین پروسسوری به حداقل رسیده است . نظریه توضیح میدهد که اتصال برای پارتیشن به حداقل رسیده است وقتی اتصال در طی هر مرز مورد نیاز است . یک نتیجه مشابه بدست می آید برای چند پردازنده های خوشه ای در [21] بعوان یک مورد خاص جالب است ملاحظه کنید مورد را . وضع نـسبت , کـه نـشان میدهـد یـک پارتیشن مربعی را همان نتایج پیدا میشود عملا در [14] و بصورت فرمان نشان داده شده اسـت در [16] .

B – پارتیشن بندی شش گوشه ای

نظریه 8 : اجازه دهید یک مجموعه الگو که را تولید میکند بعنوان اوزان ارتباط برای هر جهت . سپس به حداقل خواهد رسید هرگاه

اثبات : از همان روشهای قبلی استفاده میشود [22].

ملاحظه کنید مورد خاصی را جایی که این مورد دلالت میشود از , و که یک شش گوشه که حداقل ارتباط است . این نتیجه اثباط میشود برای هر دو صورت فرمان وعملی در [16] , و یکبار دیگر نشان میدهد چگونگی اطلاعات پارتیشن الگوریتمیک میتواند توضیح دهد موردی از ADP یک چند بعدی الحاق شده برای هر دو بهینه سازی مستطیل و شش گوشه توسعه داده شده [22] اما حذف میشود در موارد کمبود فضا.

VII – ارزشهای آزمایشی ADP

ADAPT ( ) یک برنامه پارتیشن بندی برای پیاده سازی ADP , نوشته شده است برای ترجمه یک برنامه FORTRAN به یک برنامه پیاده سازی پارتیشن مطلوب ADAPT یک ارزشیاب جزئی است که میگیرد بعنوان ورودی یک برنامه FORTRAN و پارامترهایی مثل اندازه خط cache بعنوان ورودی و تولید میکند یک برنامه FORTRAN باقاعده با ساختار موازی به Multimax . اکنون قسمتهای پارتیشن برنامه ADAPT با یک حلقه ترتیبی تکی خارجی و دو حلقه موازی خارجی است .

ADAPT تعیین میکند که مجموعه الگوها از کد برنامه تولید کند اوزان ارتباط را و برنامه را پارتیشن بندی کند .

A : یک مشکل ساده

قسمت کد داده شده در شکل 1 استفاده شده است برای آزمایش الگوی آن یک الگوی ارتباطی بدون فرم است . بنابراین پارتیشنهای استاندارد هندسی , بعنوان پیشنهاد در [13] , [16]و [14] , رو میشود برای به حداقل رساندن ارتباط . همچنین این برنامه محتوی هیچ مجموعه مستقل قابل اجرا

نیست و هیچ الگوی پارتیشن بندی تحلیلی , همچنانکه ارائه شد در [2] و [3] رو نمی شـود بـرای تعریف پارتیشنهای کاهنده ارتباط .

زمانبندی کد هدایت شده روی 16 پردازنده و یک Multimax دو پردازندهای A..N..L نـشان میدهد که یک کارائی بدست میآید توسط ADP .

در همه آزمایشات و شبیه سازیها , اندازه قسمتی که تعیین شده برای هر پردازنده انتخـاب میـشود کم میشود سپس اندازه cache برای رفع اشتباهات اجباریcache سنجش آرایه و کارایی پارتیشن انتخاب میشود برای کاهش خطاهای تصادفی .اوقات ویژه ای برای پارتیـشنهای ویـژه ای بنـابراین بازتاب میشود به همان درستی امکان پذیر ترافیک هماهنگی cache .

دومجموعه متفاوت از ارزشیابیهای آزمایشی هدایت شد برای اثبات استفاده از ADP و اولی , زمان اجرای برنامه های موازی پارتیشن بندی شده توسط ADP و استفاده از موضوعات متعارف و قابـل اندازه گیری روی Multimax 16 پردازنده ای . دومبرنامه های پارتیشن بندی شده بـا اسـتفاده از موضوعات شبیه سازی شده روی شبیه ساز اندازه گیری نسبت اشتباهات نتـایج شـبیه سـازی نـشان میدهد چگونگی استراتژی پارتیشن بندی را.

B – نتایج اجرایی

تجزیه هر بردار دسترسی داده شده در جدول I . با استفاده از روش max-min

. روی Multimax ,L=4 , با در نظرگرفتن اندازه cache 16 بایت است و اندازه اعداد شناور 4 بایت است . چون سنجشهای آرایه چند برابر L اسـت , اوزان ارتبـاط تنظـیم میشود با استفاده از (5.1) و (5.3) به و برای تطبیق اثـرات انـدازه cache . از (6.1) وضغیت نسبت پارتیشن مستطیل بهینه 0.5 است .

کارایی پارتیشنهای دینامیک و استاتیک مقایسه خواهند شد با مستطیل تولید شده توسط ADAPT . روشهای راهنمای خود زمانبندی [20] استفاده میشود بعنوان یک مثال از یک پارتیشن برنامه پویا . پارتیشنهای استاندارد استاتیک (ردیفها و فضاها) مقایسه میشود با پارتیشنهای تولیـد شـده بوسـیله ADP . مستطیلهای طراحی شده بوسیله ADAPT بهبود میبخشد زمان اجرا را به 8.2 درصد روی فضاها و 9.1 درصد روی ردیفها و 15.2 درصد روی راهنمای خود زمانبندی .

آزمایشها هدایت شد به Multimax هایی که از پارتیشنهای نـستطیلی بـا فـضای برابـر و نـسبت وزنهای متنوع . این نتایج ارائه شد در شکل 9 که میانگین زمان اجرا در میلی ثانیه طراحـی شـد در برابر نسبت وضعیت . به وضوح پارتیشنهای با نسبت عرض به طول 0.5 زمان اجرای کمتـری دارد در این زمان , بدین سان نمایش صحت الگوریتم max- min و (5.1) برای

قسمت برنامه در شکل 1 روی Multimax . تعدادکم FLOPS معین است در قسمت بطور نسبی نسبتا کم کارایی نقاط شناور Multimax است.

C – شبه سازی نتایج :

1- اندازه های مختلف cache . برای بدست آوردن بازبینی آزمایشی اثر اندازه خطای cache میتواند روی اتصال باشد . قسمت برنامه از شکل 1 اجرا شد روی یک شبه ساز یک پیکر بندی چند پردازنده ای Astronautic ZS-1 . هر پردازنده یک cache مخصوص دارد و متصل به حافظه اصلی است و از طریق یک خط عرضی همزمانی cache نگهداری شد از طریق استفاده از یک الگوی دایرکتوری محور قسمت کد اجرا شد با استفاده از پارتیشنهای نسبت متنوع وضع معکوس هنگام نگهداری نمونه اندازه پارتیشن بالا سری معین شد که اشکال پارتیشن به حداقل رساندن نسبت خطا را در اضافه این آزمایش تکرار شد برای ارزشهای خاصی از L , تعدادی از نقاط داده در طی خط cache نتایج گزارش داده شد در شکل 10 .

از یک تحلیل کد روشن کرد که . این مشخص شد یکبهینه کننده مستقل از ماشین با استفاده از (6.1) تولید میکن یک پارتیشن را با یک نسبت معین معکوس از 2 به 1 .اگرچه یک پارتیشن بند مستقل استفاده میکند از (5.2) و (3.5) برای حصول پارتیشنهای دیگری از ارزشهای دیگری از L آنچنان که در جدول II آمده است .

یک مقایسه از نسبتهای اشتباه شبیه سازی شده برای پارتیشنهای تولید شده برای استفاده از وزنهای مستقل از ماشین و وزنهای وابسته به ماشیین که بدست میدهد حداقل نسبت اشتباهات را برای یک اندازه خط جزئی.

2- شبیه سازی پارتیشنهای شش گوشه . یک پارتیشن شش گوشه ای برای یک اندازه خط جزئی باید ملاحظهکند اساس شش گوشه ای را . شش گونه ها ممکن است بر اساس افقی باشند یا آنها ممکن است بر اساس عمودی باشند , چنانکه بدست آمد بوسیله چرخش یک شش گوشه به 90 درجه .برای یک شش گوشه افقی محور , , از (4.4) بدست می آید بطور مشابه برای یک شش گوشه عمودی محور . که باید حداقل باشد.

ما قسمت کد بدست آمده در شکل 1 را استفاده می کنیم . سه پارتیشن شبیه سازی شد با استفاده از شبیه ساز چند پردازنده ای Astronautic ZS-1 با استفاده از اندازه خط 16 بایت . حل (6.4) , (6.2) و (6.3) برای یک شش گوشه ای افقی محور تعیین کرد .

. این پارتیشن یک مربع چرخیده شده به اندازه 45 درجه است . شبیه سازی برنامه پارتیشن بندی شده بوسیله این شش گوشه ها نسبت خطایی به اندازه 0.62 درصد دارند. برای شش گوشه های عمودی محور و از (7.1) . شبیه

سازی برنامه ارتیشن بندی شده بوسیله شش گوشه های منظم نسبت خطایی به انـدازه 1.31 درصـد دارند . نه آنکه استفاده از شش گوشه های بهینه شده نتیجه میدهد در یک نصف کردن نسبت خطای محاسبه شده برای استفاده از شش گوشه های منظم .

3- شبیه سازی پارتیشنها برای یک الگوی بدون فرم . در بعضی کد هموارسازی image هـا [19] الگوی (3.1) استفاده میشود برای محاسبه مشتق جزئی x و y از حرکت مرزی . این الگـو اتـصال یکتایی دارد در مرزهای افقی و عمودی , هنگام داشتن اتصال نابرابر در طی مرزهای b و d . ایـن الگوی شبیه سازی شد با اشتفاده از شبیه ساز Astronautic ZS-1 , با استفاده از خط cache بـه اندازه L=16 .

برای این مثال شبیه سازی برنامه پارتیشن بندی شده بوسیله شش گوشه های منظم نسبت خطایی دارد به اندازه 1.93 درصد . شبیه سازی برنامه پارتیشن بندی شده بوسیله شش گوشه هـای بهینـه شـده عمودی با ملاحظهپهنای باند , هنگامی که تولید شده است با اسـتفاده از (6.2) , (6.3) , (6.4) و (5.2) و (5.4) , نسبت خطایی دارد به اندازه 0.98 درصد . شش گوشه هـای بهینـه شـده بهبـود میبخشد نسبت خطای شش گوشه های منظم را به 49.2 درصد.

یک آزمایش با پارتیشنهای مستطیلی انجام شد . شبیه سازی برنامه پارتیشن بندی بوسیله مربع نسبت خطایی برابر 1.59 درصد داشت . شبیه سازی برنامه پارتیشن بندی شده بوسیله مستطیلهای بهینه شده که با استفاده از (6.1) تولید شو و (5.2) نسبت خطایی برابر 0.87 درصد دارد . مستطیلهای بهینه شده بهبود میبخشد نسبت خطای مربع را به 44.2 درصد .

مستطیلهای بهینه شده نسبت خطای کمتری دارد از شش گوشه های منظم , حتی مدل سازی *** را 150 میدهد و 141,1 . ما باور داریم که اختلاف به خاطر پارتیشنهای مستطیلی مطابقـت داده شـد بنابراین این هیچ دو قسمت تشکیل شده همان خطای cache را داشته باشند .

اشتراک سازیهای غلط اثر میکند روی ترافیک هماهنگی cache وقتی اندازه های خط بزرگ باشـد . اما اثرات اشتراک سازی فقط شمرده نمی شود برای مدل ما و نشان میدهد یک منطقه از کار را .

VIII – نتیجه گیری

این کاغذ آزمایش کرد پارتیشن بندی برنامه را بعنـوان یـک روش بـرای کـاهش زمـان اجـرای برنامههای موازی.

ارتباط بین یک حلقه موازی تکراری تقسیم شد به یک علامت بردار . بـا اسـتفاده از ایـن علامـت گزاری ارتباطات , پارتیشنهای داده با حداقل ارتباط بین پردازندهای طراحی شد با استفاده از ADP . ADP متوجه خودش است برای بودن یک قدم به جلو یک قالب کاری تئوریک بـرای تحلیـل ارتباط در برنانه های موازی روی چند پردازنده های با حافظه مشترک .

ADP یک الگوی بهینه پارتیشن بندی است که استفاده میکند از مـستطیلها و شـش گوشـه هـای نامنظم برای هر الگو . این اصولا متفاوت است با کارهای قبلی , که متمرکز بود روی توسعه بهینـه پارتیشن داده برای مجموعه الگویی خاص . در ادامه , ADP پایه گذاری شده است برای راه حلهای الگوریتمیک که مشخص شد بوسیله بررسی دقیق برنامه های ععذی , عمومی . نتـایج یـک تحلیـل خوکار بودند که توانستند کاهش دهند ترافیک همـاهنگی cache را . ADAPT یـک سـازنده خودکار پارتیشن بندی داده که اسـتفاده شـد بـرای تحلیـل برنامـه . آزمایـشات روی شـبیه سـاز Multimax و Astronautic ZS-1 اثبات کرد امتیازات ADP را . شش گوشه ها که بصورت سنتی استفاده میشوند برای پارتیشن بندی , از رده خارج شدند توسط مستطیلها.

فرصتهای زیادی وجود دارد برای کار در آینده در این مورد . محدودیت بروزرسانی ماتریس تکی قرار گرفته روی بدنه کد حلقه مـوازی تکـراری جـایگزین خواهـد شـد بـا مجموعـه عمـومی از دستورالعملها برای ساختن ADP قابل استفاده در اکثر برنامه ها . همچنین ADP میتواند توسعه پیدا کند برای ساخت دیگر برنامه سازیهای موازی و موارد خاصـی از سیـستمهای چنـد پردازنـده ای . تحلیل وسیع منجر خواهد شد به توسعه یک کلاس از کامپایلر ها که میتواننـد بـه حـداقل برسـانند ارتباط را در برنامه های موازی قابل حمل در طی یک طیف وسیع از چند پردازنده ها.