عملکرد چندپردازنده های تراشه چند رشته ای و پیامدها برای طراحی سیستم عامل

چکیده

طراحی یک سیستم عامل اغلب از معماری سخت افزار هدف تحت تاثیر قرار می گیرد. در حالی که معماری های تک پردازنده و چند پردازنده به خوبی درک شده اند، چندپردازنده های تراشه چند رشته ای (CMT) به خوبی درک نشده اند—یک نسل جدید از پردازنده های طراحی شده برای بهبود عملکرد کاربردهای حافظه-محور. فقط اولین سیستم های مجهز به پردازنده های CMT در دسترس است، به طوری که در حال حاضر درک چگونگی به دست آوردن بهترین عملکرد از چنین سیستم هایی برای ما بسیار مهم است.

هدف از کار ما, درک اصول عملکرد CMT و شناسایی مفاهیم طراحی سیستم عامل است. ما نحوه تاثیرپذیری عملکرد یک پردازنده CMT در کشاکش خط لوله پردازنده، حافظه نهان داده L1 و حافظه نهان L2 را تحلیل نموده ایم، و رویکردهای سیستم عامل برای مدیریت این منابع عملکرد حیاتی را مورد بررسی قرار داده ایم. پس از مشخص شدن این مورد که کشاکش برای کش (حافظه نهان) L2 می تواند بیشترین تاثیر منفی را بر روی عملکرد پردازنده داشته باشد، بهبود عملکرد بالقوه را تعیین نموده ایم که می تواند از برنامه ریزی سیستم عامل آگاه از-L2 به دست آید. ما یک سیاست برنامه ریزی را بر اساس اصل تعادل-تنظیم شده ارزیابی نمودیم و متوجه شدیم که این مقوله دارای پتانسیل برای کاهش نسبت های از دست رفتگی در L2 تا ۱۹-۳۷٪ و بهبود توان عملیاتی پردازنده تا ۲۷-۴۵٪ است. دستیابی به بهبود مشابه در سخت افزار به دو برابر شدن اندازه کش L2 نیاز دارد.

۱. مقدمه

یک سیستم عامل, یک لایه انتزاعی بین سخت افزار و نرم افزار فراهم می کند. کار آن, در معرض قرار دادن توان سخت افزار به کاربردها و در عین حال پنهان کردن پیچیدگی های آن است. جای تعجب نیست که معماری سخت افزار, طراحی سیستم عامل را تحت تاثیر قرار می دهد. موضوع طراحی سیستم عامل برای پردازنده های معمولی در

گذشته بررسی شده است. این مقاله با بررسی زمینه طراحی سیستم عامل برای یک خانواده از پردازنده ها شروع می شود: چندپردازنده های تراشه چند رشته ای (CMT).

پردازنده های CMT, ترکیبی از چند پردازش تراشه (CMP) و چند رشته ای سازی سخت افزار (MT) هستند- روندهای معماری امروزی طراحی شده برای بهبود مطلوبیت پردازنده با ارائه پشتیبانی بهتر در موازی سازی سطح- رشته. یک پردازنده CMP شامل چندین هسته پردازنده بر روی یک تراشه می شود که فعال شدن بیش از یک رشته را در یک زمان میسر می سازد و باعث بهبود مطلوبیت منابع تراشه می شود. یک پردازنده MT, اجرای دستورالعمل ها را از رشته های مختلف جایگذاری می کند. در نتیجه، اگر یک رشته, دسترسی به حافظه را مسدود سازد، رشته های دیگر می توانند رو به جلو پیشرفت نمایند. مطالعات متعدد, مزایای عملکرد CMP و MT 4/7] ، 13، 23، 29، 36] را نشان داده اند.

به لطف بهبود در چگالی های تراشه، ترکیب این دو رویکرد در CMTS ممکن شده است. هدف CMTS, بهبود عملکرد یک رده مهم از کاربردهای مدرن، مانند خدمات وب، سرور برنامه کاربردی، و سیستم های پردازش تراکنش بر روی خط است. این کاربردها, به خاطر استفاده ضعیف از خط لوله CPU بدنام هستند - آنها معمولاً شامل رشته های متعدد از توالی های کنترل اجرای کوتاه از عملیات های صحیح، با شاخه های مکرر پویا می شوند. چنین ساختاری, موقعیت یابی کش و دقت پیش بینی شاخه را کاهش می دهد و باعث توقفات مکرر پردازنده می شود [۱۱، ۳۰، ۳۱، ۳۲]. پردازنده های سوپر مدرن، با استفاده از اجرای سوداگرانه و خارج از دستور می توانند موازی سازی سطح-آموزش (ILP) از وظایف سنگین علمی را غصب نمایند، اما نمی توانند کار چندانی برای بارهای کاری سبک-پردازش تبادلاتی شاخه-سنگین انجام دهند. حتی برخی از معیارهای SPEC, دارای بازده های مطلوبیت خط لوله پردازنده به اندازه ۱۹٪ [۱۴] می باشند. این به این معنی است که در اکثریت زمان ها، خط لوله پردازنده استفاده نمی شود. پردازنده های CMT برای رسیدگی به این مشکل طراحی می شوند.

سیستم های مجهز به پردازنده های CMT به مرور زمان نوشتن آن, در دسترس قرار گرفته اند. آی بی ام , تراشه POWER 5 CMT خود را در تابستان سال ۲۰۰۴ [۱۸] منتشر نمود؛ Sun Microsystems و Intel برای حمل پردازنده های CMT تجاری آن در سال ۲۰۰۵ [۸، ۹] طرح ریزی نمودند.

CMTS به شیوه ای اساسی, از پردازنده های معمولی متفاوت است: آنها با ده ها زمینه رشته به طور همزمان فعال تجهیز می شوند (به عنوان مثال، پردازنده Sun's Niagara دارای هشت هسته، هر کدام با چهار زمینه رشته خواهد بود [۸]) و در نتیجه, رقابت برای منابع به اشتراک گذاشته شدید است. این کار موجب ایجاد پیامدهای جدید برای طرح های سیستم عامل هدف قرار داده شده در چنین پردازنده هایی می شود. ما نحوه تاثیر کشاکش برای خط لوله و پردازنده، حافظه نهان داده L1 و حافظه نهان L2 بر عملکرد سیستم را با این هدف مطالعه کرده ایم که درک نماییم کدامیک از این منابع به اشتراک گذاشته شده به احتمال زیاد تبدیل به تنگناهای عملکردی می شوند, به طوری که بتوانیم سیستم عامل را با جبران این تنگناها انطباق دهیم. ما دریافته ایم که زمان تاخیر ناشی از میزان ضربه ضعیف در حافظه نهان L1 می تواند به طور موثر توسط چند رشته ای شدن سخت افزاری پنهان شود، اما این کشاکش بالا برای L2 به طور قابل توجهی می تواند به عملکرد کلی پردازنده آسیب بزند. این نتیجه ما را به بررسی میزان پتانسیل در استفاده از برنامه ریزی سیستم عامل به منظور بهبود عملکرد L2 (و پس از آن به طور کلی پردازنده) سوق داد.

ما یک الگوریتم برنامه ریزی سیستم عامل را مبتنی بر اصل تعادل-تنظیم شده [۲۱] در نظر گرفته ایم و متوجه شدیم که دارای پتانسیل کاهش نسبت های گم شدن کش L2 تا ۱۹-۳۷٪ است که بازده آن, یک بهبود عملکرد ۲۷- ۴۵٪ است- بهبودی که تنها با دو برابر شدن اندازه حافظه نهان L2 در سخت افزار می توان به دست آورد.

در حالی که این نتیجه دلگرم کننده است، هنوز مشخص نیست چه کسری از این پتانسیل می تواند توسط یک اجرا تحقق یابد؛ در پایان این مقاله ما یک دستور کار پژوهش برای اجرای این الگوریتم و چالش های دخیل در انجام این کار را مورد بحث قرار می دهیم.

سهم عمده کار ما عبارتست از: تعیین کمیت اثرات کشاکش برای اجزای مختلف پردازنده های CMT بر عملکرد کلی، طراحی یک الگوریتم زمان بندی جدید که بازده عملکرد L2 را بهتر می کند، یک ارزیابی از این الگوریتم، و اقتباس از الگوی موجود برای نسبت های گم شدن به منظور عملی ساختن آن با حجم کار چند رشته ای (این کار برای ارزیابی الگوریتم زمان بندی تعادل-تنظیم شده لازم بود).

استفاده از مکانیسم های نرم افزاری برای بهترین بهره برداری از سخت افزار های موجود به ما کمک می کند تا سیستم هایی بسازیم که می توانند عملکرد خوب را با تکامل کاربردها ارائه دهند. در حالی که طراحان سخت افزار بهترین کار خود را برای کارکرد مناسب پردازنده ها با وظایف سنگین امروزی انجام می دهند، پیش بینی دقیق آینده, یک جادوگری است. از آنجا که سخت افزار نمی تواند به سرعت خود را با تغییر کاربردها وفق دهد، طراحی نرم افزار سیستم عامل که اجرای سیستم را با تکامل برنامه ها حفظ خواهد کرد, مهم است.

بقیه این مقاله به شرح زیر ساختاریافته است: بخش ۲, پیش زمینه ای در مورد سیستم های CMT را فراهم می کند، مدل CMT را توصیف می کند که ما از آن استفاده می کنیم و به بحث در مورد فن آوری شبیه سازی می پردازیم. در بخش ۳، ما منابع به اشتراک گذاشته شده بر روی یک پردازنده CMT را به منظور شناسایی تنگناهای عملکرد تجزیه و تحلیل می نماییم و نتیجه گیری می کنیم که بهینه سازی عملکرد هدفمند در کش L2 به احتمال زیاد بیشترین تاثیر را دارد. در بخش ۴, ما بهره های عملکرد بالقوه را از برنامه ریزی تعادل-تنظیم شده را برای بهبود عملکرد L2 بررسی می کنیم. ما چالش های دخیل در پیاده سازی این زمانبند را ارائه می کنیم و یک دستور کار تحقیقات آینده را در بخش ۵ طرح ریزی می نماییم. ما کار مرتبط را در بخش ۶ بحث می نماییم و در بخش ۷, نتیجه گیری می نماییم.

۲. سابقه و شبیه ساز

سیستم های بررسی شده در این مطالعه باید چندین هسته پردازنده بر روی یک تراشه (چند پردازش تراشه ای [۳۶]) و چندین زمینه رشته سخت افزار بر روی هر پردازنده (چند رشته سخت افزار) باشند.

راه های مختلفی برای پیاده سازی چند رشته ای سازی سخت افزاری وجود دارد و می توان آنها را بطور گسترده به صورت درشت دانه، دانه ریز، و به طور همزمان دسته بندی کرد. تفاوت اصلی بین این دسته بندی ها, نحوه تعویض های پردازنده میان زمینه های رشته است.

چند رشته ای سازی درشت دانه, زمانی به یک رشته تغییر می کند که یک رشته اشغال کننده بلوک های پردازنده, درخواست حافظه و یا سایر عملیات ها با زمان تاخیر طولانی [۵] را مسدود می کند. در حالی که مزایای عملکرد این معماری برای حجم کارهای چند رشته ای [۲۳] نشان داده شده است, همچنین نشان داده شده است که عملکرد در اینگونه معماری ها به واسطه هزینه بالای تعویض زمینه محدود شده است [۶].

این مسئله در مورد معماری های ریز دانه چند رشته ای یا جایگذاری شده بررسی شده است که رشته ها در هر چرخه [۶] را تعویض می کند.

معماری های چند رشته ای سازی همزمان (SMT), پشتیبانی چند زمینه ای را به پردازنده های چند مسئله، خارج از سفارش اضافه می کنند. بر خلاف پردازنده های چند-مسئله معمولی، آنها می توانند دستورالعمل ها را از جریان های دستوری مختلف در هر چرخه برای موازی سازی بهبودیافته سطح-آموزش [۷، ۱۴] صادر کنند.

مدل ما از هسته پردازشگر چند رشته ای بر اساس مفهوم چند رشته ای سازی ریز دانه (جایگذاری)، پیشنهاد شده توسط Lauden و همکاران است. [۶]. یک هسته MT دارای چندین زمینه رشته سخت افزار (معمولا دو، چهار، و یا هشت) است که در آن هر زمینه متشکل از مجموعه ای از ثبات ها و دیگر حالات رشته است. چنین پردازنده ای, اجرای دستورالعمل ها از رشته ها، سوئیچینگ بین زمینه ها در هر چرخه را جایگذاری می کند. هنگامی که یک یا چند رشته مسدود می شوند، سیستم همچنان به سوییچینگ در میان رشته های باقی مانده ادامه می دهد.

برای اهداف مطالعه ما, یک ابزار شبیه ساز سیستم CMT [۱۶] را به عنوان مجموعه ای از برنامه های افزودنی به ابزار شبیه سازی [۱۵] Simics ساختیم. Simics, شبیه سازی سیستم-کامل چند سیستم عامل سخت افزار رایج را فراهم می کند. شبیه ساز ما مبتنی بر یک ماشین UltraSPARC II® است. Simics می توانند دستگاه شبیه

سازی شده را با سیستم عامل سولاریس ™ و محیط استاندارد یونیکس خود-راه اندازی کند. همه شبیه سازی های توصیف شده در این مقاله، اجرا-محور هستند و شامل هر دو سطح کاربر و کد سیستم عامل می شود.

ابزار ما, سیستم ها را با چندین هسته CPU چند رشته ای مدلسازی می کند. تعداد هسته های CPU در هر تراشه و درجه چند رشته ای سازی سخت افزاری قابل پیکربندی هستند.

هسته پردازنده شبیه سازی شده ما دارای یک خط لوله RISC ساده با یک مجموعه از واحدهای عملکردی است. ما تصمیم گرفتیم تا یک هسته RISC کلاسیک ساده را در برابر یک پردازشگر پیچیده خارج از مرتبه شبیه سازی نماییم، زیرا ما معتقدیم که این یک معماری مناسب برای پردازنده های آینده CMT با تعداد زیادی از هسته های پردازنده بر روی یک تراشه است. ساده سازی هر هسته، قرار دادن هسته های چند رشته ای بیشتر بر روی یک تراشه را میسر می سازد - و این یک گزینه طراحی جذاب برای OLTP و حجم کار سرور است، زیرا چنین حجم کاری به طور معمول دارای درجه بالایی از موازی سازی سطح -برنامه است و از طرح های خط لوله پیچیده فوق العاده-اسکالر کمتر بهره مند می شود. علاوه بر این، یک مطالعه قبلی نشان داده است که برای چنین حجم کاری، پیچیدگی هسته باید برای افزایش تعداد زمینه های سخت افزاری [۲۴] سبک سنگین شود. علاوه بر این، ما معتقدیم که نتایج مطالعه ما که مربوط به سلسله مراتب حافظه می شود, قابل انطباق با طیف گسترده ای از معماری های چند رشته ای هستند، زیرا معماری سلسله مراتب حافظه به معماری خط لوله بستگی ندارد.

شبیه ساز ما با دقت, کشاکش خط لوله، کش L1، محدودیت های پهنای باند در اتصالات کراس بار بین کش های L1 و L2 ، کش L2، و محدودیت های پهنای باند روی مسیر بین کش L2 و حافظه شبیه سازی می کند. ما یک TLB به اشتراک گذاشته را شبیه سازی نکردیم. اندازه گیری های ما نشان داده است که TLB, یک منبع بسیار ادعاشده برای محک زنی های ما نیست. بسته به آزمایش, شبیه ساز ما با یک تا چهار هسته پردازنده، پیکربندی شده است. هر هسته شامل چهار زمینه سخت افزار، یک انبار داده KB L1۸ و یک کش دستورالعمل KB L1۱۶ (هر دو ۴ راه تنظیم ارتباطی) می شود. ما یک کش-l2 انجمنی-تنظیم شده ۱۲-راهه متحد, مشترک در میان تمام هسته ها بر روی یک تراشه را شبیه سازی نمودیم که اندازه آن بسته به آزمایش تغییر می کند. ما اندازه های کش را شبیه به

اندازه های استفاده شده در پنتیوم IV هایپر-رشته ای شده انتخاب کردیم، یک پردازنده تک هسته ای چند رشته ای که در زمان نوشتن این مقاله [۱۳] از نظر تجاری در دسترس است.

۳. منابع تنگناهای عملکرد

در این بخش, کش های خط لوله پردازنده و روی-تراشه, کش داده های L1 و کش L2 را به عنوان تنگناهای عملکردی بالقوه روی پردازنده های CMT تحلیل می نماییم. بر اساس تجربه ما, محتمل نیست که منابع مشترک ما مانند TLB یا کش دستورالعمل L1 برای بارهای کاری مورد نظر ما به تنگنا تبدیل شوند و بنابراین آنها را در این مطالعه بررسی نمی کنیم.

پردازنده های چندرشته ای به طور خاص برای پنهان نمودن زمان بیکاری حافظه تجربه شده توسط کاربردها, در زمانی که نرخ ضربه کش پردازنده ضعیف است, طراحی می شوند. هنگامی که زمان بیکاری حافظه را می توان توسط چند رشته ای نمودن سخت افزار پنهان نمود, خط لوله پردازنده به یک تنگنا تبدیل می شود. هنگامی که زمان بیکاری حافظه برای پنهان نمودن بسیار بالاست, کش های روی-تراشه به تنگنای عملکرد تبدیل می شوند.

این بخش شامل دو قسمت می شود: اول, در بخش ۳,۱, ما حالتی را در نظر می گیریم که خط لوله, تنگنای عملکرد است. ما یک تکنیک زمانبندی سیستم عامل طراحی شده برای مدیریت بهتر کشاکش خط لوله را مدیریت می کنیم. در بخش ۳,۲, ما حالتی را بررسی می کنیم که کش های پردازنده, تنگنای عملکرد هستند. به طور خاص, ما بررسی می کنیم که تحت کدام شرایط, کش داده های L1 (بخش ۳,۲,۱) یا کش L2 (بخش ۳,۲,۲) به تنگناهای عملکرد تبدیل می شود. ما نتیجه می گیریم که در حالیکه چند رشته ای سازی سخت افزاری یک کار عالی پنهان سازی زمان بیکاری ناشی از خطاها در کش داده های L1 ا انجام می دهد, توانایی آن برای پنهان نمودن زمان های بیکاری ناشی از عملکرد ضعیف در L2, بسیار بیشتر محدود می شود. در بخش ۴, ما یک الگوریتم زمانبندی سیستم عامل طراحی شده برای مدیریت کشاکش L2 را پیشنهاد می دهیم.

۳,۱ خط لوله پردازنده

رویکردهای سیستم عامل برای مدیریت کشاکش خط لوله در پردازنده های چندرشته ای قبل از [1, 2, 12] پیشنهاد شده است. با تشریح در بخش ۶, در نظر می گیریم که قابلیت کاربرد این رویکردها برای پردازنده های CMT محدود شده است, زیرا آنها به خوبی در سیستم ها با چندین زمینه نرم افزاری مقیاس بندی نخواهند شد. ما یک رویکرد با ویژگی های مقیاس پذیری بهتر را بررسی می کنیم.

مشاهده کلیدی آنها اینست که رشته ها در نحوه استفاده از خط لوله پردازنده متفاوت هستند. رشته های محاسبه-محور با اهداف شاخه قابل پیش بینی, مانند بارهای کاری علمی, دستورالعمل ها را غالباً صادر می کنند و از خط لوله به طرز شدیدی استفاده می کنند. رشته هایی که موقعیت یابی ضعیف مرجع حافظه, مانند کاربردهای پایگاه داده ها را به نمایش می گذارند, غالباً در حین انتظار برای پاسخ از سلسله مراتب حافظه متوقف می شوند: این رشته ها حافظه-محور هستند.

این مشاهده نشان دهنده نحوه مدیریت بهتر کشاکش خط لوله است. توسط برنامه ریزی همزمان رشته های محاسبه-محور جامع با رشته های حافظه-محور بر روی یک هسته پردازنده, زمانبند می تواند تقاضا برای منابع خط لوله را در سراسر هسته ها متعادل نماید. یک رشته محاسبه-محور می تواند از خط لوله در حین مسدود شدن رشته حافظه-محور در حافظه استفاده کند. این ایده شبیه به زمانبندی دسته جفت شده [۳۳] است، یک روش برای برنامه ریزی روی ابر رایانه های موازی که کارهای محاسبه-محور و I / O محور را برای بهره برداری از منبع مطلوب تطبیق می دهد.

یک زمانبند می تواند رشته های محاسبه-محور و حافظه-محور را توسط اندازه گیری معیار CPI بار کاری (چرخه ها در هر دستورالعمل) شناسایی نماید. بارهای کاری با CPIهای پایین (نزدیک ۱ روی خط لوله ساده ما) دارای کاربرد خط لوله بالاست و برعکس. با استفاده از CPI به عنوان یک ابتکار برای زمانبندی لزوماً در یک پردازنده CMT جذاب است: CPI می تواند به آسانی اندازه گیری شود, به خوبی, کاربرد خط لوله بارهای کاری را ضبط نماید و تصمیمات زمانبندی را بر اساس اطلاعات موضعی میسر سازد.

با اجرای آزمایش زیر, ما بهبود عملکرد بالقوه را از چنین سیاست زمانبندی احراز شرایط نموده ایم. ما ریزمحک ها را با CPIهای تک رشته ای زیر ساختیم. ۱, ۶, ۱۱ و ۱۶ برای شبیه سازی بارهای کاری محاسبه-محور و به طور پیشرونده حافظه-محورتر. سپس, روی یک ماشین با چهار هسته پرازنده و چهار زمینه شیار روی هر پردازنده, ما چندین روش را برای زمانبندی ۱۶ شیار آزمایش نمودیم: چهار تا هر یک با CPIهای ۱, ۶, ۱۱ و ۱۶ . برای منسوب نمودن رشته ها به یک پردازنده خاص, ما از یک فراخوانی سیستم `processor_bind()` دردسترس در Solaris استفاده نمودیم. هر رشته منسوب شده به یک هسته خاص به زمینه سخت افزاری خود محدود می شود. پردازنده, دستورالعمل های صادر شده را روی زمینه های پردازنده ه جایگذاری می کند. بنابراین تمام شیارها به صورت موازی اجرا می شوند و سهام مساوی از شیارهای صدور پردازنده را کسب می کنند. ما اندازه گیری را در حلقه اصلی محک ها آغاز می کنیم و اندازه گیری را زمانی متوقف می کنیم که هر رشته پایان یابد.

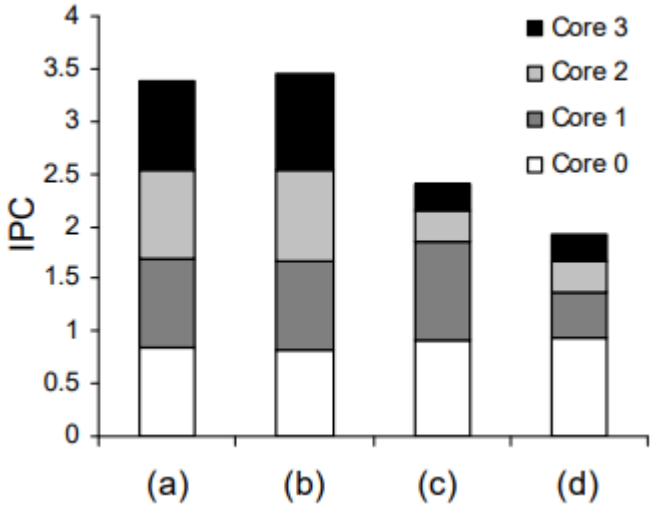
جدول ۱, چهار زمانبندی مختلف را نشان می دهد که ما ارزیابی نمودیم. زمانبندی های (a) و (b), رشته های محاسبه-محور را با رشته های حافظه-محور تطبیق می یابد و انتظار می رود که بهتر از زمان بندی های (c) و (d) عمل نماید که رشته های محاسبه-محور را روی یک هسته می گذارند.

	Core 0	Core 1	Core 2	Core 3
(a)	1, 6, 11, 16	1, 6, 11, 16	1, 6, 11, 16	1, 6, 11, 16
(b)	1, 6, 6, 6	1, 6, 11, 11	1, 11, 11, 16	1, 16, 16, 16
(c)	1, 1, 6, 6	1, 1, 6, 6	11, 11, 11, 11	16, 16, 16, 16
(d)	1, 1, 1, 1	6, 6, 6, 6	11, 11, 11, 11	16, 16, 16, 16

جدول ۱. انتساب رشته ها به هسته ها برای زمانبندی های (a)-(d). اعداد در سلول ها, CPIهای تک رشته ای رشته های منسوب شده به این هسته را نشان می دهند.

شکل ۱ نشاندهنده IPC (دستورالعمل ها در هر چرخه) به دست آمده توسط هر زمانبندی است که به واسطه CPU تجزیه می شود. طبق انتظار, زمانبندیهای a) و b), بهترین عملکرد را به دست می آورند. ۱ تفاوت در IPC بین

زمانبندی (d) و زمانبندی های (a) و (b)، یک ضریب از دو است. تجزیه IPC به واسطه CPU، نشان می دهد که دلیل عملکرد بهتر زمانبندی های (a) و (b)، اینست که آنها یک کاربرد متوازن تر منابع پردازنده را تولید می کنند. در حالیکه بهبود عملکرد بالقوه از این تکنیک زمانبندی، چشمگیر است، برای دستیابی به آن، داشتن رشته ها با گستره وسیع از CPIها لازم بود. داشتن رشته ها با گستره وسیعی از CPIها لازم بود. برای تعیین اینکه آیا بارهای کاری واقعی می توانند از این تکنیک زمانبندی به همان روشی بهره مند شوند که CPIهای تک رشته ای را برای تعدادی از معیارهای عددی استاندارد اندازه گرفتیم: SPEC JVM، SPEC CPU (int)، SPEC JBB and SPEC Web. ما یک تغییر بسیار کوچک را در سراسر معیارهای محکم دیدیم. برای بیشتر معیارهای محک، CPI حول ۴ می ماند. کوچکترین میانگین CPI که ما دیدیم ۲,۳۷ برای 164 gzip، بزرگترین- ۵,۱۱ برای 181. Mcf. بود. ترکیب دستورالعمل دینامیک نیز تغییر کمی در سراسر معیارهای محک را نشان می دهد. اگر این معیارهای محک، مشخصه بارهای کاری واقعی باشند، این داده ها نشان می دهند که تغییر کمی در استفاده از خط لوله وجود دارد. بنابراین پتانسیل کمی برای بهره های عملکرد از به کارگیری این تغییر وجود دارد. آزمایشات ما با معیارهای محک SPEC، تایید نمود که بهره های عملکرد از زمانبندی مبتنی بر CPI، برای بارهای کاری (حدود ۵٪) متوسط هستند.



شکل ۱. IPC به دست آمده توسط هر زمانبندی، تجزیه شده توسط CPU

در حالیکه مزایای زمانبندی مبتنی بر CPI برای بارهای کاری SPEC با خط لوله شبیه سازی شده ما کوچک بودند, آنها می توانند در یک سیستم SMT بزرگتر باشند. به خاطر داشته باشید که سیستم های SMT دارای چندین واحد وظیفه ای و خط لوله موضوع-متعدد هستند. گستره CPIها برای بارهای کاری عدد صحیح می توانند در چنین ماشین هایی بزرگتر باشند. چون شبیه ساز ما دارای قدرت شبیه سازی یک خط لوله سوپر-اسکالر ندارد, ما نمی توانیم تایید کنیم که این مورد وجود دارد. اما این یک حوزه جالب برای کاوش و بررسی آتی است.

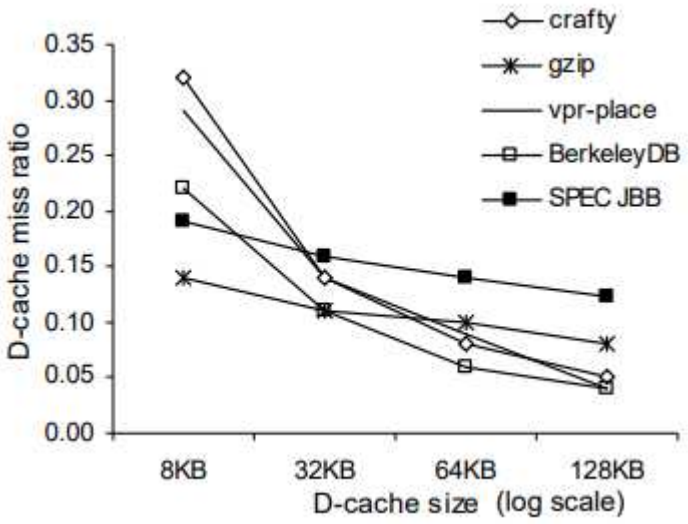
۳,۲ کش های پردازنده

۳,۲,۱ کش داده های L1

پردازنده های چندرشته ای نوعاً با کش داده های L1 کوچک پیکربندی می شوند, مثلاً کش داده های L1 روی Pentium IVهایپر-رشته ای اینتل, 8KB است [13]. ما نگران بودیم این به زمان های بیکاری بالای مرتبط با رفع خطاهای کش مرتبط می شود که پردازنده چندرشته ای نمی تواند پنهان نماید. توانایی پنهان کننده-زمان بیکار به درجه چند رشته ای شدن در هسته پردازنده بستگی ارد. ما از چهار برای درجه چند رشته ای شدن استفاده می کنیم که نشان داده شده است به بهترین شکل برای معماری ما در شبیه سازی کار [۶]. برای آزمایشات در این بخش، شبیه ساز با یک تک هسته ای پیکربندی شده است: از آنجا که حافظه نهان داده L1 روی هر هسته وجود دارد، شبیه سازی یک تک هسته ای کافی بود.

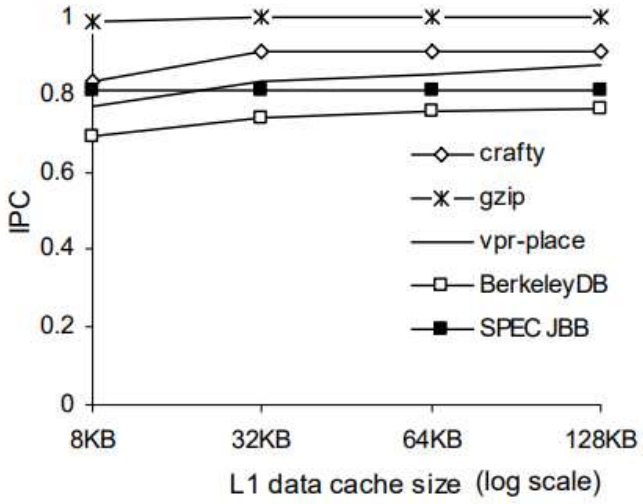
ما نسبت های گم شدن کش از دست برای اندازه های مختلف کش برای حجم کارهای زیر را اندازه گیری نمودیم: SPEC پردازنده ۱۸۶.crafty، 164.gzip، 175.vpr (مکان)، برکلی DB و تنظیمات JBB. برای هر آزمایش، چهار نسخه از یک معیار بر روی یک ماشین تک هسته ای را اجرا نمودیم – رشته های در حال اجرای حجم کار به اشتراک گذاشته کش داده هسته. معیارهای محک هیچ داده ای را به اشتراک نگذاشتند. ما با چندین اندازه کش، از ۸ KB (این می تواند اندازه کش معمولی استفاده شده بر روی یک هسته CMT باشد) تا ۱۲۸ KB آزمایش نمودیم.

همانطور که در شکل ۲ نشان می دهد، برای اندازه های کش کوچک، نسبت های گم شدن کش، روش بالا هستند که برای پردازنده های معمولی تک رشته قابل قبول در نظر گرفته می شوند.



شکل ۲. نسبت های گم شدن کش-D با تغییر اندازه کش

با این حال، همانطور که می توانیم از شکل ۳ ببینیم، عملکرد پردازنده در چنین رفتار کش ضعیف, ایمن است. حتی زمانی که نسبت های از دست رفتن کش بالاتر از ۲۰٪ هستند، IPC2 (دستورالعمل در هر سیکل) بین ۰,۷ و ۰,۹، است و با تغییرات اندازه کش, چندان تغییر نمی کند. این نشان می دهد که چند رشته ای شدن سخت افزار نشانی از یک کار خوب پنهان کردن زمان تاخیر در ارتباط با خطا در حافظه نهان داده L1 است. مفهوم این نتیجه این است که سیاست های سیستم عامل هدف یافته در بهبود عملکرد در حافظه نهان داده L1 به احتمال زیاد بهبودهای عملکرد چشمگیر را به ارمغان می آورند (و ما این را از نظر آزمایشی تایید می کنیم), زیرا سخت افزار قبلاً عملکرد کش ضعیف را می پوشاند.



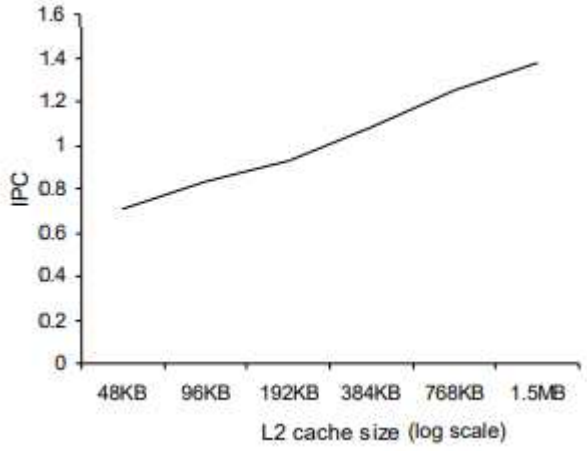
شکل ۳. مطلوبیت خط لوله با تغییر اندازه کش-D

۳,۲,۲. کش L2

کش L2 دارای پتانسیل بیشتری برای تبدیل شدن به یک تنگنای عملکرد است, زمانی که در هنگام نسبت گم شدن بالا است، به این دلیل که زمان تاخیر بین L2 و حافظه اصلی به طور قابل توجهی بیشتر از زمان بین L1 و L2 است. به عنوان مثال، در پنتیوم IV هایپر-رشته ای شده، سفر از L1 به L2, ۱۸ چرخه پردازنده طول می کشد، در حالی که سفر از L2 به حافظه اصلی ۳۶۰ چرخه [۱۳] طول می کشد. در مدل ما, این زمان شروع در ۲۰ و ۱۲۰ چرخه تنظیم شده است. جرایم از خطا کردن در L2، بیشتر به دلیل رقابت برای پهنای باند حافظه اصلی افزایش می یابند.

به منظور ارزیابی اثر عملکرد L2 بر IPC پردازنده، ما یک ماشین MT دو هسته ای با چهار زمینه رشته در هر هسته، یک کش داده ۸KB L1 L2 و انبارها با اندازه های مختلف را استفاده نمودیم.

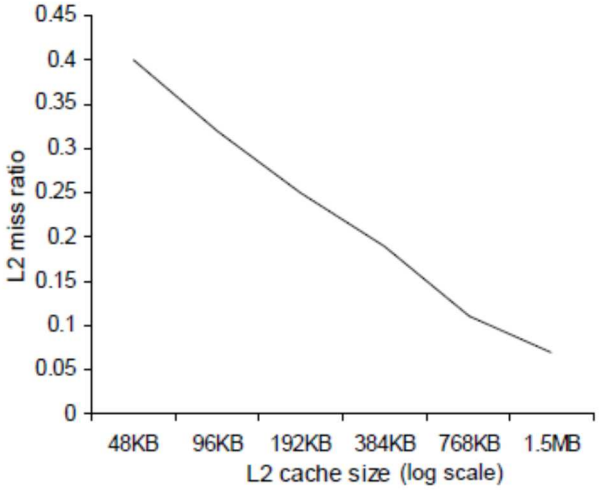
حجم کار ما شامل نه معیار از مجموعه تنظیمات CPU 2000 (۱۶۴.gzip، 175.vpr (مکان)، ۱۷۵.vpr (مسیر)، ۱۷۶.gcc، 179.art، 186.crafty، 188.ammp، 197.parser، 255.vortex) می شود. ما این معیار خاص را به این علت انتخاب کردیم که می خواستیم حجم کار ما شامل برنامه با هر دو محل کش خوب و بد باشد. ما دو نسخه از هر معیار را اجرا نمودیم که به ما حجم کار ۱۸-رشته ای را ارائه داد (هر معیار یک فرایند تک رشته ای است که در رشته خود اجرا می شوند).



شکل ۴. IPC برای بار کاری SPEC ۱۸- فرآیند. پردازنده IPC, به اندازه L2 حساس است.

همانطور که در بخش ۲ توضیح داده شده، شبیه سازی های ما اجرا-محور هستند، و معیارهای محک در محیط عملیاتی سولاریس ™ ۹ اجرا می شوند. سولاریس ™ ۹ پشتیبانی ویژه برای پردازنده های CMT را شامل نمی شود. از نقطه نظر سیستم عامل، ماشین چهار راه چند رشته ای دو هسته ای شبیه سازی شده ما مثل یک هشت راه معمولی چندرشته ای به نظر می رسد. زمانبند سولاریس ™ , رشته ها را به زمینه های سخت افزار اختصاص می دهد, مثل اینکه آنها را به پردازنده ها بر روی یک سیستم چندپردازنده منسوب می نماید و هشت رشته را در یک زمان برای زمانبندی در هشت زمینه سخت افزار در طول هر زمان برنامه ریزی شده می گنجاند. ما اندازه حافظه نهان L2 را تغییر می دهیم و نشان می دهیم که چگونه این کار بر عملکرد L2 و IPC پردازنده تاثیر می گذارد. پنتیوم IV هایپر-رشته ای دارای یک کش ۲۵۶ KB L2 است. ما هر دو کش های کوچک و بزرگ را برای کشف اثرات کشاکش کش سبک و سنگین بر عملکرد، بدون نیاز به تغییر حجم کار شبیه سازی نمودیم. ما به سرعت شبیه سازی را تا اجرای تمام رشته ها برای رسیدن به حلقه اصلی اجرا نمودیم و سپس یک شبیه سازی دقیق برای یک میلیارد چرخه را انجام دادیم.

شکل ۴, IPC پردازنده حاصل از اجرای این حجم کار SPEC را نشان می دهد. تخریب عملکرد با کوچک شدن حافظه نهان L2 آشکار است. شکل ۵ نشان می دهد که دلیل تخریب عملکرد نسبت از دست رفتن بالا برای اندازه L2 کوچک است.



شکل ۵. نسبت های گم شدن L2 برای بار کاری SPEC ۱۸-فرآیند. نسبت گم شدن L2, با افزایش اندازه L2 افت می کند.

این نتیجه نشان می دهد که توانایی پردازنده های چند رشته ای برای پنهان کردن تاخیر بالای ناشی از رفع خطاها در L2 محدود شده است. از آنجا که کاربردهای مدرن, روند خطرناکی از داده محورتر بودن را نشان می دهد، ممکن است که پردازنده های CMT مجهز به یک کش به اندازه کافی بزرگ L2 برای وظایف سنگین امروزی قادر به برآوردن نیازهای کاربردهای در آینده ای نزدیک نخواهد بود. از آنجا که تغییر نرم افزار آسان تر از تغییر سخت افزاری است، تجهیز سیستم عامل با توانایی رسیدگی به کمبود منابع در هنگامی که سخت افزار نتواند این کار را انجام دهد عاقلانه است. در بخش بعدی, کمیت بهبود عملکرد بالقوه که از برنامه ریزی سیستم عامل بهتر به دست می آید, قابل دستیابی است.

۴. برنامه ریزی BALANCE SET

برنامه ریزی تعادل-تنظیم شده توسط Denning [۲۰] به عنوان یک روش بهبود عملکرد حافظه مجازی پیشنهاد شده است. ما اثربخشی این رویکرد را برای کش L2 بررسی نمودیم. ایده پشت برنامه ریزی تعادل-تنظیم شده به شرح زیر است. تمام رشته های اجرایی در زیر مجموعه ها را جدا کنید، و یا گروه ها، به طوری که مجموعه کاری ترکیبی هر گروه متناسب با کش است. بنابراین، یک گروه را در یک زمان را برای مدت زمان برنامه ریزی برش زمانی

برنامه ریزی کنید. با اطمینان از اینکه مجموعه کاری از هر گروه برنامه ریزی شده متناسب با حافظه نهان است، این الگوریتم به منظور کاهش نسبت های گم شدن کش طراحی شده است.

با این حال، ما متوجه شدیم که اندازه تنظیم شده کاری, یک شاخص خوب رفتار کش حجم کاری نیست. هنگامی که ما از یک مدل بر اساس اصل تنظیم شده کاری [۲۲] برای برآورد نسبت های از دست رفتن کش وظایف سنگین SPEC استفاده نمودیم، برآوردها اغلب متناقض بودند: یک حجم کار با یک مجموعه کاری بزرگ, نسبت های از دست رفتن کمتر از یک حجم کار با مجموعه کاری کوچک را تولید خواهد کرد، و بالعکس. تحقیقات بیشتر [۳۹] نشان داد که مجموعه کار, یک شاخص خوب از نسبت از دست رفتن کش حجم کار است, تنها در صورتی که برنامه به مجموعه کاری خود به طور یکنواخت دسترسی پیدا کند - این فرض دسترسی یکنواخت, فرض اساسی کار اصلی دنینگ در برنامه ریزی تعادل-تنظیم شده [۲۱] است. همانطور که ما یاد گرفتیم، این فرض برای معیارهای استاندارد برقرار نیست (CPU SPEC، وب SPEC، SPEC JBB).

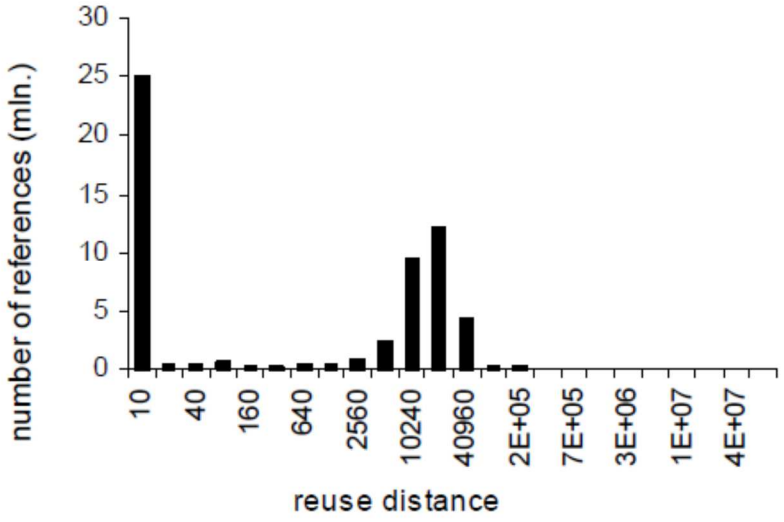
ما به پیدا کردن یک راه بهتر برای ارزیابی رفتار کش حجم کار نیاز داریم، به طوری که ما یک مدل را جستجو نمودیم که به دقت نسبت های از دست رفتن کش را بر اساس برخی از ویژگی های حجم کار برآورد می کند. در بخش ۴,۱ ما یک مدل مورد استفاده برای نسبت از دست رفتن کش و نحوه سازگاری این مدل را توصیف می کنیم به طوری که بتوان آن را با برنامه ریزی تعادل-تنظیم شده استفاده نمود. در بخش ۴,۲, کارایی بالقوه الگوریتم زمان بندی تعادل-تنظیم شده بر اساس این مدل مورد بررسی قرار می دهد.

۴,۱. مدلی برای نسبت از دست رفتن کش

Berg و Hagersten یک مدل برای برآورد نسبت از دست رفتن کش بر اساس فواصل استفاده مجدد [۱۹] را توسعه دادند. یک فاصله استفاده مجدد, مقدار زمانی است که بین مراجع پی در پی به یک محل از حافظه سپری می شود. این مدل, احتمال یک ضربه کش در فاصله استفاده مجدد را پایه گذاری می کند. هرقدر فاصله استفاده مجدد کوچکتر باشد، این احتمال بیشتر است که مرجع به یک ضربه منجر شود. با استفاده از این مدل، Berg و Hagersten, نسبت از دست رفتن کش وظایف سنگین SPEC در حدود ۵ درصد مقادیر واقعی برآورد نمودند.

شکل ۶ یک مثال از یک هیستوگرام فاصله-استفاده مجدد (توزیع غیر نرمال فاصله-استفاده مجدد) برای ammp.۱۸۸ را نشان می دهد. نسبت قابل توجهی از فواصل استفاده مجدد بزرگ هستند - این نCOMB ای از محل ضعیف است. آزمایش های ما تایید کردند که این معیار محک زنی استثنائاً دارای نسبت از دست رفتن کش فوق العاده بالا هستند.

ورودی لازم برای این مدل، یک هیستوگرام فاصله-استفاده مجدد، در زمان اجرا با نظارت بر منابع حافظه، با استفاده از مکانیسم سخت افزار نقطه تماشا استاندارد [۱۹] قابل ساختن است.



شکل ۶. توزیع فاصله استفاده مجدد برای 188.ammp.

مدل فاصله-استفاده مجدد, نسبت از دست رفتن کش برای حجم کار را تخمین می زند. برای برنامه ریزی تعادل-تنظیم شده، باید تعیین نماییم که رشته ها, پایین ترین نسبت از دست رفتن را زمانی تولید می کنند که برای اجرای با هم و دسترسی به کش به صورت همزمان برنامه ریزی شوند. بنابراین، ما باید نسبت از دست رفتن کش برای گروه های رشته ها را برآورد نماییم. برای انجام این کار، ما نمودار هیستوگرام فاصله-استفاده مجدد فردی را برای رشته ها می سازیم و سپس این نمودار هیستوگرام ها را به منظور برآورد نسبت از دست رفتن برای گروه از رشته ها ترکیب می نماییم.

ما سه روش برای برآورد نسبتهای از دست رفتن گروهی توسعه دادیم: COMB, COMB+IPC, and AVG.

COMB ترکیبی از هیستوگرام های فاصله-استفاده مجدد در یک هیستوگرام تک است و از این هیستوگرام برای

پیش بینی نسبت از دست رفتن برای گروه استفاده می کند. IPC + COMB در COMB توسط تنظیمات در تفاوت های نسبی در سرعت های رشته های در حال اجرا بهبود می دهد. AVG, نسبت های از دست رفتن برای هر موضوع در یک گروه را برآورد می کند چنانکه با یک بخش اختصاص داده شده از حافظه نهان کوچکتر اجرا می شود و سپس نسبت های حاصل را میانگین گیری می نماید. ما در حال حاضر این روش ها را با جزئیات شرح می دهیم و سپس دقت و صحت آنها را ارزیابی می کنیم.

۴,۱,۱. روش # ۱: COMB

برای ترکیب چندین هیستوگرام ها در یک هیستوگرام، از الگوریتم دو مرحله ای زیر استفاده می کنیم:

۱. برای هر فاصله استفاده مجدد، تعداد منابع قرار گرفته در این فاصله را برای همه هیستوگرام ها جمع کنید.

۲. مقدار هر فاصله استفاده مجدد ظاهر شده در هیستوگرام توسط N یا N-1 را که در آن N تعداد هیستوگرام های ترکیب شده است (به عنوان مثال، تعداد رشته ها) ضرب می کنیم. شکل ۷ را ببینید.

مرحله ۲ ضروری است زیرا زمانی که چندین رشته, کش را با استفاده از چند رشته ای سازی ریز دانه به اشتراک می گذارند، فاصله استفاده مجدد برای هر ارجاع دوباره حافظه افزایش خواهد یافت. برای مثال، اگر یک رشته در حال اجرا روی یک مرجع دوباره خود باشد, محل حافظه در درون یک گام زمان، هنگامی که با سه رشته دیگر اجرا می شود، قبل از آن که به آن مکان حافظه ارجاع کند، رشته دیگر می تواند با ارجاع به حافظه خود دخالت کند. در نتیجه، فاصله استفاده مجدد اصلی به عنوان مثال، می تواند در هر نقطه بین یک تا چهار باشد. ما با تنظیم فاصله استفاده مجدد توسط ضرب آن در ضرایب از ۱ تا N آزمایش نموده ایم که در آن N تعداد هیستوگرام ها (تعداد رشته ها) ترکیب شده است. ما دریافتیم که استفاده از یک ضریب N-1 بهتر برای کش های بزرگتر بهتر کار می کند، و ضریب N برای کش های کوچکتر بهتر کار می کند.

Histogram A	
R.dist	**# ref.**
1	90
10	50
20	78
…	…
100	14

Histogram B	
R.dist	**# ref.**
1	30
10	46
20	27
…	…
100	18

Histogram A+B	
R.dist	**# ref.**
2	120
20	96
40	105
…	…
200	32

شکل ۷. ترکیب هیستوگرام های فاصله-استفاده مجدد

روش بعدی ما، IPC + COMB ، قطعاً این ضریب را با در نظر گرفتن نرخ های مختلف که در آن رشته ها, منابع حافظه را صادر می کنند محاسبه می کند.

۴,۱,۲. روش ۲: IPC + COMB

وقتی که چند رشته, یک کش را به اشتراک می گذارند، الگوهای دسترسی کش در رشته های کندتر تاثیر کمتری بر نسبت از دست رفتن کش کل نسبت به رشته های سریع خواهند داشت، چرا که رشته های کندتر, منابع حافظه با سرعت کندتر را صادر خواهند کرد. در نظر گرفتن این تفاوت ها در زمان ترکیب هیستوگرام های فاصله-استفاده مجدد لازم است.

نرخ حافظه مرجع، تعداد مراجع در هر چرخه، را می توان به دو مولفه تجزیه نمود، منابع در آموزش و دستورالعمل در هر سیکل (IPC):

$$Refs.\ per\ cycle = Refs\ per\ instr.\ * IPC.$$

برای حساب کردن تفاوت های نسبی در مراجع در هر دستورالعمل، ما داده ها را برای استفاده مجدد از هیستوگرام های فاصله روی یک پنجره از دستورالعمل ها برای تمام رشته ها جمع آوری می کنیم.

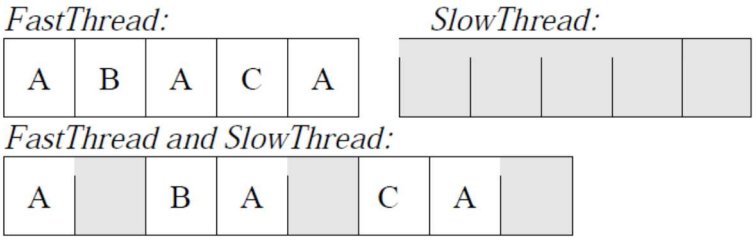
برای درک نحوه تنظیم تفاوتهای نسبی در دستورالعمل در هر سیکل (IPC)، مثال زیر را در نظر بگیرید:

فرض کنید FastThread دو برابر سرعت SlowThread اجرا می شود- IPC های مربوطه آنها ۱ و ۰,۵ می باشند. هنگامی که این دو رشته با هم اجرا می شوند، درحدود نیمی از مراجع FastThread با مراجع

SlowThread پراکنده خواهند شد. بنابراین، برای FastThread، نصف فاصله استفاده مجدد, یکسان باقی خواهند ماند و نیمی دیگر توسط یک ضریب دو افزایش می یابد. (نگاه کنید به شکل ۸.) برای منعکس کردن این مورد، فواصل استفاده مجدد در هیستوگرام FastThread را در ضریب زیر ضرب نمودیم:

(۰,۵ * ۱ + ۰,۵ * ۲) = ۱,۵

با این حال، SlowThread، چندان خوش شانس نخواهد بود: همه منابع حافظه آن با مراجع ساخته شده توسط FastThread پراکنده خواهند شد. در واقع، هر مرجع که SlowThread را می سازد با دو ارجاع به حافظه ساخته شده توسط FastThread پراکنده خواهند شد. بنابراین، تمام فاصله استفاده مجدد با ضریب سه افزایش می یابد.



شکل ۸. توضیح نحوه تغییر فواصل استفاده مجدد در زمانی که رشته ها با سرعت های نابرابر به طور موازی اجرا می شوند.

این مشاهدات ما را به فرمول DIST_COEFF هدایت می کنند. ضریبی که به واسطه آن ما مقادیر فواصل استفاده هر رشته را تنظیم می نماییم.

$$DIST\_COEFF_i = \left(\sum_{j=1}^{n} IPC_j \right) / IPC_i ,$$

این مشاهدات ما را به فرمول زیر برای DIST_COEFF هدایت می کنند, ضریبی که به واسطه آن ما مقادیر فواصل استفاده مجدد هر رشته را تنظیم می کنیم:

$$DIST\_COEFF_i = \left(\sum_{j=1}^{n} IPC_j \right) / IPC_i ,$$

که در آن i شاخص رشته است که ضریب آن را محاسبه نمودیم و n تعداد کل رشته ها در این گروه است.

در نهایت، ما نیاز به ایجاد یک تنظیم برای تعداد مراجعی داریم که در داخل هر فاصله-استفاده مجدد قرار می گیرند: یک رشته کند,منابع حافظه را آهسته تر از رشته سریع صادر خواهد کرد، بنابراین ما نیاز به مقیاس تعداد مراجع داریم که بر این اساس در هر فاصله-استفاده مجدد قرار می گیرند. این باعث تولید فرمول زیر برای محاسبه REFS_COEFF می شود- ضریب مقیاس بندی متناظر:

$$REFS\_COEFF_i = \mathrm{IPC}_i / \mathrm{MAX\_IPC},$$

که در آن i, شاخص رشته است که ضریب آن را محاسبه کردیم و MAX_IPC بزرگترین IPC در ترکیب رشته ها است.

هنگامی که ما هر هیستوگرام فاصله استفاده مجدد را با ضرب فاصله استفاده مجدد در DIST_COEFF و تعداد ارجاعات توسط REFS_COEFF تنظیم نمودیم، به سادگی تمام هیستوگرام های فاصله-استفاده مجدد (شکل ۹) را ادغام نمودیم.

Histogram A (IPC = 1)	
R.dist	**# ref.**
1.5	90
15	50
30	78
…	…
150	14

Histogram B (IPC = 0.5)	
R.dist	**# ref.**
3	15
30	23
60	14
…	…
300	9

Histogram A+B	
R.dist	**# ref.**
1.5	90
3	15
15	50
…	…
300	9

شکل ۹. ترکیب هیستوگرام های فاصله-استفاده مجدد تنظیم شده. توجه داشته باشید که ما از همان هیستوگرام های مثال شکل ۷ استفاده مودیم و تنظیماتی را برای آنها با توجه به روش شرح داده شده در این بخش صورت دادیم.

اشکال یک رویکرد مبتنی بر ترکیبی مانند این است که پیاده سازی یک سیستم واقعی بیش از حد از نظر محاسباتی پر هزینه است. یک ماشین با ۳۲ زمینه رشته و ۱۰۰ رشته را تصور کنید. زمانبند به طور بالقوه $\binom{100}{32}$ هیستوگرام

را ترکیب می کند و نسبت از دست رفتن را برآورد می کند. اگر چه این محاسبات اغلب انجام نمی شوند (همانطور که در بخش ۶ توضیح داده شده است)، هنوز هم به سربار اضافه می کند. رویکرد توصیفی بعدی ما عملی تر است.

۴,۱,۳. روش ۳: AVG

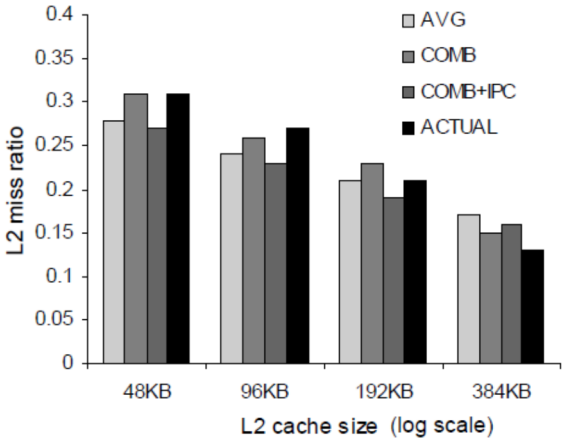
شهود پشت AVG, تظاهر این مورد است که هر رشته با پارتیشن اختصاصی کش خود اجرا می شود. AVG, نسبت های از دست رفتن برای هر رشته را برآورد می نمایند چنانکه با یک کش کوچک تر، با اندازه TOTAL_CACHE / NUMBER_OF_THREADS اجرا می شود. سپس، برای پیش بینی نسبت از دست رفتن در یک گروه از رشته ها، نسبت های تخمینی رشته های فردی را میانگین گیری می نماییم. ما دریافتیم که پیش بینی ها بر اساس این روش درست به اندازه پیش بینی های مبتنی بر روش های مبتنی بر ترکیب دقیق هستند و به میزان قابل توجهی در تولید کمتر گران قیمت هستند.

برآورد نسبت از دست رفتن باید یک بار برای هر حجم کار محاسبه شود، و پس از آن، نسبت از دست رفتن برای وظایف سنگین متعدد تصویر شود. زمانبند تنها باید چندین مقدار را متوسط گیری نماید. علاوه بر این، زمانبند می تواند میانگین ها برای گروه های کوچک از رشته ها را به یاد آورد، و سپس نسبت های از دست رفتن برای یک گروه بزرگتر را با ادغام میانگین های شناخته شده پیش بینی نماید.

۴,۱,۴. ارزیابی دقت مدل

به منظور آزمایش اثربخشی مدل، از ۱۸ معیار حجم کار SPEC CPU2000 در بخش ۳,۲,۲ استفاده می کنیم. ما نسبت های از دست رفتن برای تمام گروه های ممکن متشکل از هشت رشته، برای چهار اندازه حافظه نهان L2 را با استفاده از COMB، COMB + IPC و AVG برآورد می نماییم. ما نسبت های از دست رفتن تخمین زده شده برای آن گروه از رشته ها را اعتبارسنجی می نماییم که بهترین زمانبندی برای الگوریتم زمان بندی تعادل-تنظیم شده را برای هر اندازه کش تولید می نماییم. (ما توضیح می دهیم که چگونه این گروه ها در بخش بعدی انتخاب شدند.)

برای اعتبارسنجی نسبت های از دست رفتن برآورد شده برای گروه، ما هر گروه بر روی یک ماشین را با دو هسته و چهار زمینه سخت افزار در هر هسته برای ۱۰۰ میلیون سیکل بعد از انتقال سریع شبیه سازی بعد از مرحله مقدار دهی اولیه معیار محک شبیه سازی نمودیم.



شکل ۱۰. نسبت های از دست رفتن پیش بینی شده در برابر واقعی

شکل ۱۰, نسبت های از دست رفته پیش بینی شده و واقعی به دست آمده با استفاده از سه روش برای اندازه های کش مختلف را نشان می دهد. نسبت های از دست رفتن نمایش داده شده, کمیت های میانگین گیری شده برای گروه های رشته تولید کننده بهترین زمانبندی برای هر اندازه کش (انحراف استاندارد, کوچک است). نسبت های از دست رفتن برآورد شده ما, به طور متوسط, درون ۱۷٪ نسبت های از دست رفتن واقعی هستند. خطاها, انتظار می روند, با توجه به اینکه ما مدل را برای یک بار کاری چند-برنامه ای به کار بردیم, یک کش ارتباطی-تنظیم شده شبیه سازی نمودیم (مدل استفاده مجدد-فاصله, یک کش ارتباطی-کامل را فرض می کند) و از یک اندازه پنجره ۵۰۰ میلیون دستورالعمل را در زمان جمع آوری داده ها برای هیستوگرام های فاصله-استفاده مجدد استفاده نمودیم (Berg و Hagesten, استفاده از پنجره های کوچکتر را به منظور اجتناب از فازهای گسترش مختلف در رفتار یک برنامه پیشنهاد می دهند).

هرچند نرخ های خطا بالا هستند, کاهش آنها برای ما حیاتی نیست: آنچه مهم است اینست که مدل برای تمایز بین گروه های رشته ها به اندازه کافی دقیق است که نسبت های از دست رفتن بالا را تولید می کند و مدل هایی که

نسبت های از دست رفتن کم را می سازند. و همانطور که ما در بخش بعدی نشان دادیم, این دقیقاً چیزی است که ما برای بهبود عملکرد زمانبند پیش فرض نیاز داریم.

۴,۲ زمانبندی تنظیم شده-تعادل

قابلیت پیش بینی نسبت های از دست رفتن کش در گروه های رشته ها, شناسایی گروه های رشته ای را ممکن می سازد که نسبت های از دست رفتن کش پایین را تولید می کند. ما این مورد را در الگوریتم زمانبندی توازن-تنظیم شده اعمال می کنیم. قبل از پیاده سازی الگوریتم, می خواستیم بهبود عملکرد را تعیین نماییم که از استفاده از آن انتظار می روند. این بخش توصیف می کند که چگونه ما پتانسیل این الگوریتم را برای کاهش کشاکش L2 و بهبود عملکرد پردازنده توسط شبیه سازی اقدامات آن ارزیابی می کنیم. اولاً ما الگوریتم زمانبندی را توصیف می کنیم و سپس توضیح می دهیم که چگونه ما آن را شبیه سازی می کنیم. در بخش ۴,۳, نتایج عملکرد و تحلیل را ارائه می دهیم. ما چالش های پیاده سازی و هزینه های زمان اجرای بالقوه در بخش ۵ را بررسی می کنیم.

۴,۲,۱ الگوریتم زمانبندی

مرحله ۱ – انتخاب آستانه نسبت از دست رفتن L2 (که به صورت متناوب انجام می شود, همانند مجموعه رشته های در حال اجرا روی تغییرات سیستم):

شغل زمانبند, حفظ نسبت های از دست رفتن L2 در زیر این آستانه خواهد بود. هدف, تنظیم آستانه نسبت از دست رفتن تا حد ممکن پایین است, اما نه چندان کم که برخی رشته ها را تغذیه نکند. زمانبند از همان اول, حریص ترین رشته کش را شناسایی می کند, زیرا این رشته است که وارد خطر گرسنگی می شود. نسبت های از دست رفتن به دست آمده از تحلیل هیستوگرام های فاصله-استفاده مجدد فردی برای شناسایی این رشته استفاده می شوند.

بعد، زمانی که زمانبند نسبت های از دست رفتن برای تمام گروه های رشته را تحلیل کرده است، نسبت های از دست رفتن برآورد شده برای گروه را بررسی می کند که شامل حریص ترین رشته می شود و کوچکترین نسبت از دست رفتن را انتخاب می کند. این کوچکترین نسبت از دست رفتن که می تواند بدون گرسنه نگهداشتن حریص ترین رشته به دست آید، و این نسبت از دست رفتن است که به عنوان آستانه انتخاب می شود.

مرحله ۲ - تجزیه و تحلیل نسبت از دست رفتن برای شناسایی گروه های رشته که نسبت های از دست رفتن کش کم را تولید خواهد کرد (به صورت تناوبی انجام می شود به عنوان مجموعه رشته های در حال اجرا در تغییراتی در سیستم):

از حوزه کل رشته های اجرایی، گروه های رشته را بسازید، به طوری که تعداد رشته ها در هر گروه برای تعداد زمینه های سخت افزاری در دسترس می باشند. نسبت های از دست رفتن L2 برای تمام گروه ها را با استفاده از مدل فاصله-استفاده مجدد و روش هیستوگرام-ترکیب AVG برآورد نمایید. آن گروه هایی که نسبت از دست رفتن برآوردشده بالاتر از آستانه است را کنار بگذارید. گروه های باقی مانده, گروه های نامزد هستند.

مرحله ۳ - تصمیم برنامه ریزی (هر بار که یک زمان برنامه ریزی تکه منقضی می شود, انجام می شود):

یک گروه از مجموعه گروه های نامزد را انتخاب نمایید. رشته ها در گروه را برای اجرا در طول برش زمان کنونی اجرا کنید. (توجه داشته باشید که یک رشته ممکن است به بیش از یک گروه تعلق داشته باشد.) میزان زمان پردازنده هر رشته را پیگیری نمایید. در برش زمانی بعد، یک انتخاب از گروه های نامزد صورت دهید که حاوی رشته هایی می شوند که قبلاً کم و یا هیچ زمان پردازنده را دریافت نکرده اند.

۴,۲,۲. تقلید زمانبند

در این بخش ما ابتدائاً توضیح می دهیم که چگونه فرآیند تصمیم گیری زمانبندی را شبیه سازی می کنیم و سپس نحوه شبیه سازی زمانبندی حاصل را توضیح می دهیم. ما بار کاری SPEC ۱۸ رشته توصیف شده در بخش ۳,۲,۲ و پردازنده CMT با دو هسته و چهار زمینه سخت افزاری را روی هر هسته توصیف می کنیم. ما شبیه ساز خود را برای جمع آوری داده ها در هیستوگرام های فاصله-استفاده مجدد ابزاربندی نمودیم و هیستوگرام های فاصله-استفاده مجدد را در رشته ها به صورت آفلاین ساختیم.

۴,۲,۲,۱. تصمیم گیری برنامه ریزی اول،

اولاً, ما تمام $\binom{18}{8}$ ترکیبات از رشته ها را بررسی می کنیم.

- ما در مجموع ۱۸ رشته داریم, اما تنها هشت تا را می توان در زمینه های سخت افزاری هشت پردازنده در یک زمان زمانبندی نمود. ما نسبت های از دست رفتن گروهی را برای تمام گروه ها محاسبه می کنیم و گروه ها را در مرتبه صعودی توسط نسبت از دست رفتن دسته بندی می کنیم. سپس, آستانه نسبت از دست رفتن را به صورت توصیف شده در مرحله ۱ از الگوریتم زمانبندی انتخاب می کنیم (بخش ۴,۲,۱). گروه هایی که نسبت از دست رفتن برآورد شده آنها کمتر از آستانه است, مجموعه کاندید را تشکیل می دهند.

از مجموعه کاندید, چندین گروه را انتخاب می کنیم که زمانبندی نهایی را تشکیل می دهند. چون هر گروه دارای هشت رشته است و بار کاری کلی دارای ۱۸ رشته است, یک زمانبندی باید شامل حداقل ۳ گروه باشد تا اطمینان حاصل شود که هر یک از ۱۸ رشته اجرا می شوند. ایده این کار اینست که هر گروه در یک زمانبندی برای یک بخش زمانی زمانبندی عمل می کنند؛ زمانی که همه گروه ها در زمانبندی اجرا شوند, زمانبندی تکرار می شود.

برای انتخاب گروه های رشته در یک زمانبندی, دو سیاست را آزمایش نمودیم: عملکرد گرا (PERF)، و انصاف گرا (FAIR).

با PERF، از مجموعه کاندید, گروه با کمترین نسبت از دست رفتن و حاوی رشته های هنوز انتخاب نشده را انتخاب می کنیم. ما دوباره این کار را تکرار می کنیم تا هر رشته در برنامه ارائه شود.

با FAIR، ما برای تساوی سهم CPU هر حجم کار تلاش می کنیم: زمانی که ما گروه های رشته ها را انتخاب می کنیم, دنبال می کنیم که هر رشته چند بار انتخاب شده است. هر زمان که ما یک انتخاب را صورت دادیم، گروهی را انتخاب می کنیم که شامل بیشترین تعداد از حداقل رشته های اغلب انتخاب شده می شود.

ما این سیاست ها از نظر عملکرد و انصاف در بخش ۴,۳ مقایسه می کنیم. ما فرآیند زمانبندی-ساخت بالا را برای چهار اندازه حافظه نهان L2 از KB۴۸ تا KB۳۸۴ تکرار می کنیم.

۴,۲,۲,۲. شبیه سازی

هنگامی که ما برنامه های زمانبندی برای تمام اندازه کش و سیاست ها را می سازیم، هر برنامه را برای اندازه گیری عملکرد آن شبیه سازی می نماییم. برای یک برنامه معین، هر یک از گروه های آن را برای یک قطعه زمانی (۱۰۰

میلیون چرخه) بعد از ارسال سریع شبیه سازی پس از فاز مقداردهی معیار محک شبیه سازی می نماییم. برای کسب IPC در زمانبندی, IPCهای به دست آمده توسط گروه ها در آن زمانبندی را متوسط گیری می نماییم (انحراف استاندارد, کوچک است, نوعاً درون ۵٪ میانگین). بدین ترتیب, ما نسبت از دست رفتن L2 را برای زمانبندی محاسبه می کنیم.

به خاطر داشته باشید که هر گروه شامل هشت رشته می شود- یکی برای هر زمینه سخت افزار روی ماشین. با اجرای بار کاری هشت رشته CPU-محور روی یک ماشین با هشت زمینه سخت افزاری و با هیچ چیز در حال اجرا روی آن ماشین, تضمین می کنیم که هر شیار, روی زمینه سخت افزاری آن اجرا می شود. بدین ترتیب, عمل یک زمانبندی واقعی را شبیه سازی می نماییم که به طور خالص هر رشته را به زمینه سخت افزاری آن در یک قطعه زمانی منسوب می نماییم.

چون هر رشته به زمینه مناسب آن منسوب شد و دستگاه, دستورالعمل ها را از هر زمینه در یک حالت راند-رابین صادر نمودیم, تضمین نمودیم که تمام رشته های شبیه سازی شده توسط یک گروه, سهام برابر معین شیارهای موضوع پردازنده بودند و پیشرفت مستقیم را صورت دادند.

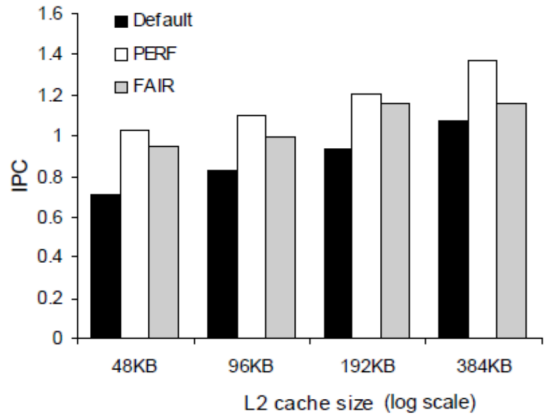
برای اندازه گیری نسبت از دست رفتن IPC و L2 در زمانبند پیش فرض, بار کاری ۱۸-معیار را روی ماشین شبیه سازی شده در ۱ میلیارد چرخه بعد از ارسال سریع معیارها بعد از فزار مقداردهی اجرا نمودیم. رشته ها به صورت دینامیکی به زمینه های سخت افزاری توسط زمانبند سولاریس منسوب شدند (توضیح بخش ۳,۲,۲ را به خاطر آورید)

۴,۳ نتایج

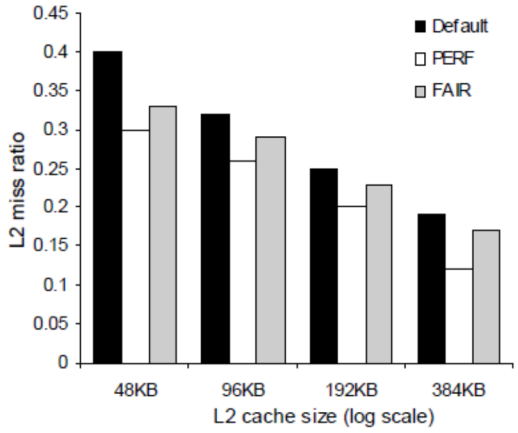
شکل ۱۱ نشاندهنده IPC برای هر زمانبندی است. در تمام موارد, زمانبند تعادل-تنظیم شده بهتر از زمانبند پیش فرض عمل می کند. بهره عملکرد از زمانبند با استفاده از سیاست PERF, ۲۷٪ (384KB L2) تا ۴۵٪ (48KB L2) است. با سیاست FAIR, بهره عملکرد از (384KB L2) 7% تا (48KB L2) 34% است. زمانی که فشار کش بزرگتر شود, منفعت بیشتر از رویکرد زمانبندی تعادل-تنظیم شده درو می شود. با کاهش کشاکش کش, مزیت

عملکرد کمتر است؛ اگر عملکرد به دست آمده با زمانبند پیش فرض از قبل خوب باشد, فضای کمی برای بهبود وجود دارد.

شکل ۱۲, نشاندهنده نسبت های از دست رفتن L2 متناظر است. با زمانبندی تعادل-تنظیم شده, در زمان استفاده از سیاست PERF, قادر به کاهش نرخ های از دست رفتن L2 تا ۱۹-۳۷٪ هستیم و در زمان استفاده از سیاست FAIR به اندازه ۸-۱۹٪. قرار دادن این نتیجه به واسطه ارزیابی معانی که برای کسب بهبودهای مشابه در سخت افزار لازم می باشد, جالب است. با کش 48KB L2, زمانبندی PERF تعادل-تنظیم شده, نسبت از دست رفتن L2 ۱۲٪ را به دست می آورد. برای کسب نسبت از دست رفتن مشابه با زمانبند پیش فرض, اندازه کش L2 باید 96K باشد- دو برابر بزرگتر. این نسبت برای اندازه های کش باقیمانده برقرار است.



شکل ۱۱. IPC به دست آمده با زمانبند پیش فرض, و زمانبند تنظیم شده-تعادلی با استفاده از خط مشی های PREF و FAIR



شکل ۱۲. نسبت های از دست رفتن حافظ نهان L2 به دست آمده با زمانبند پیش فرض و زمانبند تنظیم شده-تعادلی با استفاده از خط مشی های PREF و FAIR.

بهبود عملکرد که ما از این حقیقت به دست آوردیم که گروه های رشته را ساخته ایم, به طوری که آنها کش را از روی محبوبیت به اشتراک می گذارند که موجب تولید یک نسبت از دست رفتن L2 پایین و کسب IPC بالا به عنوان یک نتیجه می شود. تحلیل نسبت از دست رفتن ما که قبل از ساخت زمانبندی بود, به ما در اجتناب از گروه های زمانبندی کمک نمود که آشغال سازی و چرخه های پردازشگر زباله را روی سفرهای بی انتها به حافظه القا نمودند. در حالیکه این ناشن می دهد که زمانبندی تعادل-تنظیم شده دارای پتانسیل کاهش کشاکش کش و بهبود عملکرد است, پرداختن به سوالات زیر مهم است: آیا این نتایج در هر بار کاری قابل دستیابی هستند؟ آیا انصاف باید قربانی شود؟ ما بررسی از این سوالات را در بخش زیر فراهم می کنیم.

۴,۳,۱ بررسی

۴,۳,۱,۱ بار کاری

ما از معیارهای SPEC CPU برای آزمایشات استفاده نمودیم, زیرا یک بار کاری استاندارد مورد استفاده برای ارزیابی عملکرد CPU است. ما اعتقاد داریم که این بار کاری برای آزمایشاتی مناسب است که بر سلسله مراتب حافظه تاکید دارند, زیرا این مجموعه معیارها از نسخه های قبلی, به طور خاص برای این هدف بهبود یافته است [3], گنجاندن برنامه هایی که ردپاهای حافظه بسیار بزرگتر از اندازه های کش سنتی هستند اصلاح شده اند.

بار کاری ما شامل معیارهای محک با موقعیت کش خوب (164.gzip, 197.parser) و نیز موقعیت کش (188.ammp, 179.art) می شود, بنابراین گروه های رشته حاصل دارای عملکرد کش متغیر بودند. اگر بر خلاف این, تمام رشته ها در بار کاری یکسان باشند, تمام گروه های رشته, عملکرد کش یکسان را تولید می کنند. در این مورد, زمانبند باید, رشته های کمتر را نسبت به زمینه های سخت افزاری دردسترس زمانبندی نماید تا فشار روی L2 کاهش دهند. هرچند این یک شمشیر دولبه است: در حین اجرا, رشته های کمتر موجب بهبود عملکرد در L2 می شود که زمینه های سخت افزاری استفاده نشده ممکن است بر عملکرد آسیب بزنند.

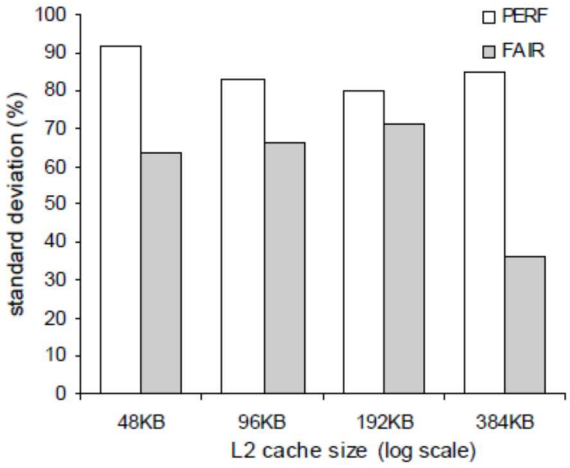
از تجربه ما, بازدهی از مبادله عملکرد بهتر در L2 برای زمینه های سخت افزاری استفاده نشده, به بار کاری وابسته است است. به منظور تصمیم گیری زمانبند در مورد زمان انجام این کار, باید پیش بینی نحوه تاثیر عملکرد L2 بر IPC پردازنده میسر شود. ما یک مدل را توسعه داده ایم که این [34] را انجام می دهد و ما برای گنجاندن آن در پیاده سازی زمانبند آینده برنامه ریزی می نماییم.

۴,۳,۱,۲ انصاف

انصاف موازنه برای عملکرد یک موضوع مرکزی در رویارویی با طراحی الگوریتم های زمانبندی است [35,38]. به منظور به دست آوردن نسبت از دست رفتن L2 پایین, کار زمانبندی تعادل-تنظیم شده, انتخاب گروه هایی از رشته هاست که نسبت از دست رفتن برآورد شده آنها, کمترین است. چنین گروه هایی احتمالاً شامل رشته های کش صرفه جو بیشتر از رشته های کش حریصانه می شود. در نتیجه, رشته های کش صرفه جو می توانند سهم بالاتری از CPU به دست آورند. به طور مثال, در آزمایشات ما, یک معیار محک صرفه جو- کش 197.parser در تقریباً هر گروه شبیه سازی گنجانده شد, در حالیکه یک 188.ammp حریص-کش تنها در یک یا دو گروه در هر زمانبندی حاضر بود.

هرچند پرداختن به انصاف زمانبندی تعادل-تنظیم شده, هدف این مطالعه نبود, ما برای بهبود انصاف با سیاست رشته-انتخاب FAIR تلاش نمودیم.

برای ارزیابی نحوه مقایسه PERF از نظر انصاف توسط FAIR, درجه انصاف به دست آمده توسط این سیاست ها را با استفاده از معیار زیر برآورد نمودیم: انحراف استاندارد از سهم CPU متوسط. تحت یک زمانبند سهم-منصفانه, هر بار کاری, سهمی برابر از CPU را به دست می آورد و انحراف استاندارد صفر است.



شکل ۱۳. ارزیابی انصاف برای دو خط مشی زمانبندی

شکل ۱۳, انحرافات استاندارد برای PERF و FAIR را نمایش می دهد. هرچند در هنگام استفاده از سیاست FAIR, زمانبند تا حدودی منصفانه تر از زمان استفاده از PERF است, انحراف استاندارد از متوسط برای سیاست FAIR نسبتاً بالاست.

سطح انصاف به دست آمده در آزمایشات ما برای زمانبندی تعادل-تنظیم شده بنیادی نیست. این خاص بار کاری ماست. به طور مثال, اگر ما یک بار کاری داشتیم که در آن تعداد رشته های کش-صرفه جو بسیار بیشتر از تعداد رشته های کش-حریص بود, همیشه تطبیق یک رشته حریص-کش با رشته های کش-صرفه جو بدون قربانی نمودن انصاف ممکن بود. رابطه بین بار کاری و انصاف قابل دستیابی, یک موضوع غنی است که ما برای بررسی آن در آینده برنامه ریزی نموده ایم.

مقدار انصاف قابل قبول برای قربانی نمودن به خاطر عملکرد, خاص هر سیستم است. در سیستم هایی که در آنها هر انصاف قربانی شده, قابل قبول نیست, زمانبندی تعادل تنظیم شده, سیاست زمانبندی انتخاب نخواهد بود به طور محتمل, یک زمانبند سیستم خواهان قدرت در تصمیم گیری در مورد میزان انصافی است که برای سبک سنگین

کردن در عملکرد می خواهند که وابسته به اولویت های نسبی رشته ها در بار کاری آن و به اهداف عملکرد تنظیم شده برای سیستم خواهد بود. بنابراین, تجهیز زمانبند با مکانیزم های کمک در چنین تصمیم گیری مهم است. کار ما روی اثرات مدلسازی نسبت از دست رفتن کش روی IPC پردازنده [34], یک گام در این جهت است.

۵. پیاده سازی زمانبند

پیاده سازی الگوریتم زمانبندی تنظیم شده-تعادل, تحت کار مداوم است. در این بخش, چالش های دخیل در این وظیفه را بررسی می کنیم و دستور کار تحقیقاتی مرتبط را بررسی می کنیم.

عملیات های زمانبند که بیشترین کمک را به سربار اجرا می کنند, مجموعه داده ها برای هیستوگرام های فاصله-استفاده مجدد و تحلیل نسبت از دست رفتن هستند.

هیستوگرام های فاصله-استفاده مجدد باید زمانی ساخته شوند که رشته های جدید وارد سیستم می شوند و سپس دوباره ساخته می شوند, زمانی که شیارهای موجود, الگوهای دسترسی کش خود را تغییر می دهند. جمع آوری داده ها در هیستوگرام های فاصله-استفاده مجدد را می توان با نظارت بر موقعیت های حافظه تحت دسترسی توسط یک رشته با استفاده از مکانیزم نقطه تماشای سخت افزاری انجام داد. Berg و Hagersten, یک ابزار سطح کاربر را توصیف می کنند که هیستوگرام های فاصله-استفاده مجدد را بدین ترتیب می سازد [19]. آنها سربار زمان اجرای کمتر از ۲۰٪ را برای کابردهای اجرای-طویل گزارش می دهند. رفع تله های کرنل مرتبط با نقاط تماشا, برجسته ترین منبع سربار است که بیشتر آنها در صورتی حذف می شوند که نظارت در کرنل انجام شود. تنها یک نمونه از موقعیت های حافظه باید نظارت شوند. کاهش نرخ نمونه برداری موجب کاهش سربار نظارتی می شود, اما می تواند به دقت کاهش یافته مدل منجر شود. ما برای بررسی توازن درست بین این دو برنامه ریزی نمودیم.

میزان ذخیره مورد نیاز برای هیستوگرام های فاصله-استفاده مجدد در حدود ۱۰۰ بایت در هر هیستوگرام است. گستره فواصل-استفاده مجدد ممکن می تواند بسیار بزرگ باشد, بنابراین به منظور کاهش الزامات فضایی, هیستوگرام ها را با جمع نمودن فواصل-استفاده مجدد در باکت ها فشرده نمودیم. پیش بینی های ما دقیق ماند حتی با اینکه از کمتر از ۲۰ باکت استفاده نمودیم.

تحلیل نسبت از دست رفتن با استفاده از روش AVG مرتبط با تخمین نسبت های از دست رفتن برای رشته های منفرد و سپس میانگین گیری کمیت ها در گروه های رشته ها می شود. این عملیت ها باید با وارد یا ترک شدن رشته های جدید به سیستم و در زمان به روزسازی هیستوگرام های فاصله-استفاده مجدد در رشته های موجود انجام شوند. برای اجتناب از نسبت های از دست رفتن در تمام گروه های ممکن رشته ها, برای طراحی یک الگوریتم حریص برنامه ریزی می کنیم که یک مجموعه کاندید را بعد از تحلیل نسبت های از دست رفتن در تعداد کمی از گروه های رشته می سازد. ما برای ارزیابی هزینه این فرآیند در نرخ های مختلف ورود و خروج رشته ها در این سیستم برنامه ریزی می نماییم.

در هنگام طراحی الگوریتم ها برای سیستم های عملیاتی چندپردازنده, اجتناب از پیاده سازی هایی که نیازمند داده های کلی هستند, حیاتی است, زیرا این می تواند به آمیختگی چنین داده هایی در میان کش های L1 پردازنده ها منجر شود که منجر به زمان های بیکار بالا می شود [26,37]. ما باید این اصل طراحی مهم را در هنگام پیاده سازی زمانبند تنظیم شده-تعادل اتخاذ نماییم. از سوی دیگر, این ممکن است که این کار در پردازنده های CMT چندان مهم نباشد که در آن زمان بیکاری مرتبط با خطا در L1, توسط چند رشته ای شدن سخت افزار پنهان می شود.

۶. کار مرتبط

کار قبلی, الگوریتم های زمانبندی را برای پردازنده های تک هسته ای SMT پیشنهاد داده است که موجب بهبود زمان پاسخ سیستم تا ۱۷٪ شده اند [1,2,12]. این الگوریتم ها شامل نمونه برداری فضای زمانبندی های ممکن و استفاده از زمانبندی هایی می شوند که به بهترین شکل عمل می کنند. این روش می تواند به طور مجازی بدون سربار پیاده سازی شود, اما به حمایت سخت افزاری نیاز دارد. الگوریتم زمانبندی ما به این علت متفاوت است که از مدلسازی برای پیش بینی بهترین زمانبندی استفاده می کند. مدلسازی برای نمونه برداری در هنگام بزرگ بودن فضای نمونه ترجیح داده می شود, مثلاً روی یک سیستم مجهز شده با چندین زمینه رشته (یعنی یک پردازنده CMT) که صدها رشته را ایجاد می کند. مقایسه اثربخشی و هزینه های روش پیشنهادی در مطالعه SMT برای ما

روی یک پیکربندی CMT بزرگ جالب خواهد بود. امیدواریم ایده ها از مطالعه پیگیرانه در مورد گنجاندن اولویت ها در زمانبند آگاه از SMT [2] را برای بهبود انصاف زمانبند خود اعمال نماییم.

ما مدل فاصله-استفاده مجدد را برای برآورده نسبت های از دست رفتن کش در رشته هایی که به طور همزمان به یک کش دسترسی دارند, اتخاذ نمودیم. یک روش جایگزین قبل از [27] پیشنهاد شده است. روش ها ما دقیق هستند اما کمتر از روش های گران از نظر محاسباتی.

یک مطالعه مرتبط توسط Thekkath و Eggers, رشته های زمانبندی همزمان را بر اساس الگوهای به اشتراک گذاری داده های آنها در نظر گرفتند [28]. هرچند, این انتظار که قرار دادن رشته هایی که عملکرد در کش L1 را بهبود می بخشند, شهودی است, این مطالعه نشان داد که چنین سیاست زمانبند, منافع عملکردی چشمگیری را ارائه نمی دهد.

زمانبندی منسجم [10], یک زیرساخت زمانبندی برای کاربردهای سرور است که اجرای عملیات های مشابه از درخواست های مختلف را دسته بندی می کند. این کار موجب بهبود محل داده داده, افزایش IPC پردازنده تا ۳۰٪ و کاهش از دست رفتن های کش L2 تا ۵۰٪ می شود. قابلیت کاربرد این تکنیک به یک رده خاص از کاربردها محدود می شود (البته یک رده مهم), و تغییرات چشمگیر برای کد کاربرد را نیاز دارد.

بسته رشته Capriccio [17], زمانبندی آگاه از منبع را با نظارت بر رفتار رشته ها پیاده سازی می کند و الزامات منبع آنها را اندازه گیری می کند. این اطلاعات در تصمیم گیری های زمانبندی برای بهینه سازی مطلوبیت منبع استفاده می شود. این روش کلی تر از روش ماست که در آن می تون, استفاده از انواع مختلف منابع را بهبود بخشید, اما به نظارت نسبتاً مفصل از حالت برنامه نیاز دارد

۷. نتایج

در این مقاله, ما نتایج اولین مطالعه ارزیابی عملکرد یک پردازنده CMT را ارائه نمودیم. ما تحلیل نمودیم که چگونه کشاکش برای خط لوله پردازنده, کش L1 و L2 بر عملکرد تاثیر می گذارد. تعیین نمودیم که کشاکش برای کش L2

دارای بیشترین تاثیر بر عملکرد سیستم است- بنابراین, این جاییست که طراحان سیستم باید بر تلاش های بهینه سازی آنها تمرکز نمایند.

ما بررسی نمودیم که چگونه زمانبند سیستم عامل بر کاهش فشار بر کش L2 با استفاده از زمانبندی تنظیم-تعادل تاثیر می گذارد. برای کار زمانبندی تعادل-تنظیم شده, مدل کش فاصله-استفاده مجدد را برای برآورد نسبت های از دست رفتن در رشته هایی که به طور همزمان به کش دست می یابند اتخاذ نمودیم.

نشان دادیم که با زمانبندی تعادل تنظیم شده, کاهش نسبت از دست رفتن کش L2 تا ۱۹-۳۷٪ و افزایش عملکرد تا ۲۷-۴۵٪ ممکن است. با این حال, بهبود عملکرد می تواند به از دست رفتن انصاف منجر شود.

برای تعیین این مورد که آیا زمانبندی تعادل-تنظیم شده برای سیستم های واقعی بادوام است یا خیر, آن را پیاده سازی نمودیم و سربار زمان اجرای آن را ارزیابی کردیم. همچنین برای بررسی تاثیر مشخصات بار کاری در بهره های عملکرد بالقوه از این الگوریتم و موازنه های انصاف مرتبط برنامه ریزی می نماییم.

8. REFERENCES

[1] A. Snavely, D. Tullsen, "Symbiotic Job Scheduling for a Simultaneous Multithreading Machine," *ASPLOS IX*, 2000.

[2] A. Snavely, D. Tullsen, G. Voelker, "Symbiotic Jobscheduling with Priorities for a Simultaneous Multithreading Processor," *SIGMETRICS*, 2002.

[3] SPEC CPU2000 Web site: http://www.spec.org/cpu2000/analysis/memory/

[4] R. Alverson et al., "The Tera Computer System", *Proc. 1990 Intl. Conf. on Supercomputing*.

[5] A. Agarwal, B-H. Lim, D. Kranz, J. Kubiatowicz, "APRIL: A Processor Architecture for Multiprocessing", *ISCA*, June 1990.

[6] J. Laudon, A. Gupta, M. Horowitz, "Interleaving: A Multithreading Technique Targeting Multiprocessors and Workstations", *ASPLOS VI*, October 1994.

[7] J. Lo, S. Eggers, J. Emer, H. Levy, R. Stamm, D. Tullsen, "Converting thread-level parallelism into instruction-level parallelism via simultaneous multithreading", *ACM TOCS 15, 2*, August 1997.

[8] Jonathan Schwartz on Sun's Niagara processor: http://blogs.sun.com/roller/page/jonathan/20040910#the_difference_between_humans_and

[9] Intel web site, http://www.intel.com/pressroom/archive/speeches/otellini20030916.htm

[10] J. Larus, M. Parkes, "Using Cohort Scheduling to Enhance Server Performance", *USENIX Tech. Conf.*, June 2002.

[11] J. Lo et al., "An Analysis of Database Workload Performance on Simultaneous Multithreaded Processors", *ISCA*, June 1998.

[12] S. Parekh, S. Eggers, H. Levy, J. Lo, "Thread-sensitive Scheduling for SMT Processors", http://www.cs.washington.edu/research/smt/

[13] N. Tuck, D. Tullsen, "Initial Observations of the Simultaneous Multithreading Pentium 4 Processor", *PACT*, September 2003.

[14] D. Tullsen, S. Eggers, H. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism", *ISCA*, June 1995.

[15] P. Magnusson et al., "SimICS/sun4m: A Virtual Workstation", *USENIX Tech. Conf.*, June 1998.

[16] D. Nussbaum, A. Fedorova, C. Small, "The Sam CMT Simulator Kit.", *Sun Microsystems TR 2004-133*, March 2004.

[17] R. von Behren, J. Condit, F. Zhou, G. C. Necula, E. Brewer, "Capriccio: Scalable Threads for Internet Services," *SOSP*, October 2003.

[18] "IBM eServer iSeries Announcement", http://www-1.ibm.com/servers/eserver/iseries/announce/

[19] E. Berg, E. Hagersten, "Efficient Data-Locality Analysis of Long-Running Applications," TR 2004-021, University of Uppsala, May 2004

[20] P. Denning, "The working set model for program behavior", *CACM 11*, 5 (May 1968), 323-333.

[21] P. Denning, "Thrashing: Its causes and prevention", *Proc. AFIPS 1968 Fall Joint Computer Conference*, *33*, pp. 915-922, 1968.

[22] A. Agarwal, M. Horowitz, J. Hennessy, "An Analytical Cache Model," *ACM TOCS 7*, pp. 184--215, 1989.

[23] R. Eickenmeyer et al., "Evaluation of multithreaded uniprocessors for commercial application environments", *ISCA'96*.

[24] L. Barroso et al., "Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing", *ISCA'00*.

[25] J. Torrellas, A. Tucker, A. Gupta, "Evaluating the performance of cache-affinity scheduling in shared-memory multiprocessors", *Journal of Parallel and Distributed Computing 24*, pp. 139—151, Feb. 1995.

[26] F. W. Burton and M. R. Sleep, "Executing Functional Programs on a Virtual Tree of Processors", *Proceedings of the ACM FPCA, Portsmouth, NH*, pp. 187-194, ACM, 1981.

[27] G. E. Suh, S. Devadas, and L. Rudolph, "Analytical cache models with application to cache partitioning," *15th International Conference on Supercomputing*, 2001.

[28] R. Thekkath, S. Eggers, "Impact of Sharing-Based Thread Placement on Multithreaded Architectures", *ISCA'94*.

[29] J. M. Borkenhagen, R. J Eickemeyer, R. N. Kalla, S.R. Kunkel, "A multithreaded PowerPC processor for commercial servers", *IBM Journal of Research and Development 44*, 6, pp. 885.

[30] A. Ailamaki, D. DeWitt, M. Hill, D. Wood. "DBMSs on modern processors: Where does time go?" *VLDB '99*, September 1999.

[31] A. Barroso, K. Gharachorloo, E. Bugnion, "Memory System Characterization of Commercial Workloads", *ISCA'98*.

[32] K. Keeton, D. Patterson, Y. He, R. Raphael, and W. Baker, "Performance characterization of a Quad Pentium Pro SMP using OLTP Workloads", *ISCA'98*.

[33] Y. Wiseman, D. Feitelson, "Paired Gang Scheduling", *IEEE TPDS*, *14*, 6, pp. 581-592, 2003.

[34] A. Fedorova, M. Seltzer and M. Smith, "Modeling Effects of Memory Hierarchy Performance on the IPC of Multithreaded Processors", *Harvard* University Technical Report, In Preparation. Contact fedorova@eecs.harvard.edu for a copy.

[35] M. Seltzer, P. Chen, J. Ousterhout, "Disk Scheduling Revisited", *Proceedings of the USENIX Winter 1990 Technical Conference*

[36] K. Olukotun, B.Nayfeh, L. Hammond, K. Wilson and K. Chang, "The Case for a Single-C Multiprocessor", *ASPLOS* 1996.

[37] B. Gamsa, O. Krieger, J. Appavoo, M. Stun "Tornado: Maximizing Locality and Concurre in a Shared Memory Multiprocessor Operat System." *OSDI 1999*, pp. 87-100

[38] D. Ellard and M. Seltzer, "NFS Tricks Benchmarking Traps", *Proceedings of USENIX Annual Technical Conference, FREE Track*, June 2003

[39] A. Fedorova, M. Seltzer, C. Small and Nussbaum, "Throughput-Oriented Scheduling Chip Multithreading Systems", *Technical Rep TR-17-04*, Harvard University, August 2004.