ملاحظات طراحی سیستم عامل های پردازنده شبکه

چکیده

پردازنده های شبکه (NP) یک زیرساخت پردازش بسته قابل برنامه ریزی و انعطاف پذیری برای سیستم های شبکه وعده داده اند. برای بهره برداری کامل از قابلیت های پردازنده های شبکه، توانایی انطباق پویا با الگوهای ترافیک در حال تغییر و پشتیبانی بلادرنگ در قالب یک سیستم عامل پردازنده شبکه ضروری است. تفاوت های بین سیستم عامل های موجود و چالش های اصلی در ماهیت چندپردازنده ای NPها، محدودیت منابع on-chip آنها و الزامات پردازش بلادرنگ نهفته است. در این مقاله، مسائل اصلی که باید در طراحی یک سیستم عامل پردازنده شبکه درنظر گرفته شوند، بررسی میگردند. به طور خاص، تاثیر عملکردی (1) تجزیه و تحلیل برنامه برای پارتیشن بندی (2) خصوصیات ترافیکی شبکه، (3) نگاشت حجم کار و (4) سازگاری زمان اجرا کاوش میشوند. نتایج کمی و کیفی در زمینه تجزیه و تحلیل یک نرم افزار و نگاشت یک چارچوب خاص ارائه و مورد بحث قرار گرفته، اما به طور کلی مشاهدات و نتیجه ها در هر محیط برای پردازنده های شبکه قابل اجرا می باشند.

دسته بندی ها و توصیف موضوع: D.4.1 [سیستم های عامل] چندپردازنده ای؛

کلمات کلیدی: پردازنده شبکه، پارتیشن بندی برنامه، نگاشت نرم افزار

1. معرفی

موفقیت اینترنت به عنوان یک رسانه ارتباطی در اجرای تحقیقات در حوزه شبکه های حسگر، شبکه های همپوشان، محاسبات فراگیر، پردازش توری و شبکه های ذخیره سازی اطلاعات می باشد. این روند قابلیت شبکه را در جهت

دربرداشتن پروتکل ها و سرویسهای سیستم های پایانی بطور فزاینده متنوع و ناهمگن گسترش می دهد. حتی در اینترنت امروزه، روترها حجم زیادی پردازش را در مسیر داده انجام میدهند. نمونه هایی از این قبیل عبارتند از دیوارهای آتش، ترجمه آدرس شبکه (NAT)، سوئیچینگ وب، ردیابی IP، TCP/IP سیل آسا برای سرورهای ذخیره سازی با کارایی بالا و رمزگذاری شبکه های خصوصی مجازی (VPN) . بسیاری از این توابع در شبکه های دسترسی و edge اجرا میشوند که متنوع ترین سیستم ها و توابع مورد نیاز شبکه را به نمایش می گذارند. با گسترش دامنه شبکه می توان انتظار داشت که این روند ادامه خواهد داشت و پردازش پیچیده تر بسته های داخل شبکه ضرورت می یابد [2، 4].

زیرساخت پردازشی برای این عملیات پردازش بسته مختلف را می توان در راه های زیادی پیاده سازی نمود. وظایف خوش تعریف، با سرعت بالا اغلب در مدارهای مجتمع با کاربرد خاص (ASIC ها) اجرا شده اند.

وظایفی که به خوبی تعریف نشده و یا احتمالا در طول زمان تغییر میکنند را باید در یک پلت فرم انعطاف پذیرتر پیاده سازی کرد که توانایی برنامه ریزی دوباره داشته باشد. پردازنده های شبکه (NP) برای این منظور توسعه داده شده اند.

تقاضای عملکردی افزایش سرعت لینکها و نیاز به انعطاف پذیری نیازمند این واقعیت است که این پردازنده های شبکه در سیستم های چند پردازنده پیاده سازی شوند. این امر برنامه نویسی چنین دستگاه هاییی دشوار باشد، ازآنجا که عملکرد کلی به تعامل خوب اجزای مختلف سیستم (پردازنده ها، رابط های حافظه، ساختمان داده های اشتراکی و غیره) بستگی دارد. مشکل اصلی رسیدگی به پیچیدگی ها و تعامل اجزای مختلف سیستم NP است. برای دستیابی به عملکرد پردازشی لازم برای پشتیبانی از لینک های چند گیگابیتی، پردازنده های شبکه به عنوان سیستم های چند پردازنده روی یک تراشه (on-a-chip) پیاده سازی شده اند. این شامل چندین موتور پردازش چند رشته ای، انواع مختلف حافظه درون و بیرون از تراشه (on-off-chip) و تعدادی پردازنده خاص منظوره است. در ایستگاه های کاری معمولی یا سیستم های سرور این پیچیدگی ها توسط سیستم عامل پنهان شده و یا با توجه به معماری تک پردازنده به شدت ابراز نمیشوند. برای تسهیل این روند، در حال حاضر تعدادی از زبان های برنامه نویسی دامنه

خاص و کامپایلرهای بهینه سازی در حال توسعه اند [5، 17، 19]. هدف این رویکردها ، بهینه سازی یک اپلیکیشن واحد (به عنوان مثال عملیات روتر) بطور استاتیک برای سخت افزار زیر بنایی می باشد.

در پردازنده های شبکه فعلی، بسیاری از تسک های حیاتی عملکردی و خوش تنظیم با زبان اسمبلی نوشته شده اند (مثلا برای توازن زمان پردازش هر مرحله در یک خط لوله نرم افزار). در نتیجه، تغییرات جزئی در عملکرد می تواند تاثیر عملکردی شدیدی داشته باشد که به تنظیم دوباره نیاز دارد. به علت تنظیم دقیق برنامه های منفرد، یکپارچه سازی و تغییر پویای چند تابع پردازش بسته بر روی یک پردازنده تک بسیار دشوار است. با این حال، پردازش شبکه ذاتا یک فرایند پویا می باشد. انگیزه اصلی پیاده سازی توابع پردازش بسته در یک پردازنده شبکه (به جای یک دستگاه منطقی سفارشی سریع تر و کارآمدتر) لزوم تغییر عملکرد در طول زمان است. تغییر الگوهای ترافیکی، سرویس ها و پروتکل های شبکه جدید، الگوریتم های جدید برای طبقه بندی جریان و تغییر دفاع در برابر حملات انکار سرویس، زمینه پویایی ارائه میکند که یک روتر قابل برنامه ریزی نیاز به اصلاح داشته باشد. این مستلزم آن است که روتر (1) بتواند چند برنامه پردازش بسته را در همان زمان پیاده سازی کند (2) بتواند به سرعت توابع پردازشی را به حجم کار خود اضافه و حذف کند و (3) بتواند عملیات کارآمدی را تحت هر شرایطی تضمین نماید. به ویژه مدیریت منابع مختلف سیستم برای جلوگیری از تضعیف عملکرد منابع گلوگاهی حائز اهمیت است.

پیچیدگی معماری چند پردازنده NP استفاده از مفاهیم سیستم عامل های موجود در این حوزه را محدود کرده است. چند روش موجود برای پشتیبانی بلادرنگ هنوزهم به سوی برنامه های پردازش بسته واحدی سوق دارند که بصورت آفلاین بهینه سازی شده [19، 12] و یا برنامه های پردازش شبکه را به عنوان یک نهاد یکپارچه واحد در نظر میگیرند [9].در این مقاله، ما انواع مختلفی از مسائل مربوط به طراحی یک محیط بلادرنگ را کاوش میکنیم که از چند برنامه پردازش شبکه همزمان پشتیبانی می کند و اجازه پیکر بندی دوباره حجم کار را بر روی یک سیستم چندپردازنده می دهد. ملاحظات طراحی کلیدی مورد بحث در چهار دسته عمده قرار می گیرند: (1) پارتیشن بندی نرم افزار (2) خصوصیات ترافیکی (3) نگاشت و سازگاری زمان اجرا و (4) محدودیت های سیستم.

تلاش های تحقیقاتی در حوزه پردازنده شبکه در طولانی مدت توسط طراحی های سیستمی خاص تحت تاثیر قرار گرفته اند. به منظور جداسازی مسائل مربوط به طراحی عمومی سیستم های عامل از یک سیستم خاص، ما مشاهدات کیفی و نتایج کمی را از هم تفکیک کرده ایم. ابتدا مسائل مربوط به طراحی ذکر شده فوق بررسی شده و در مورد مبادلات کیفی بین گزینه های طراحی موجود بحث میشود. این تجزیه و تحلیل درک مهمی از تعامل بین مبادلات سیستم فراهم می کند. ثانیا نتایج کمی ارائه میشود که مبادلات فوق در زمینه برنامه های واقعی NP را به تصویر میکشد. برای این کار، از پارتیشن بندی نرم افزار NP و تکنیک های نگاشت و نتایج مدل سازی عملکرد تحلیلی که در کار قبلی ما توسعه داده شد [15، 20] استفاده میشود.

در بخش 2، درمورد کارهای مرتبط بحث میشود. بخش 3 ویژگی های کلیدی سیستم عامل های پردازنده شبکه و تفاوت آنها از سیستم عامل های ایستگاه های کاری رایج مطرح میگردد. بخش 4 مبادلات طراحی کیفی را مورد بحث قرار داده و بخش 5 این مشاهدات را در زمینه سیستم مورد آزمایش ما بصورت کمی تخمین میزند. در بخش 6 نتایج و پیامدهای طراحی پردازنده شبکه سیستم عامل ترکیب شده و بخش 7 این مقاله را نتیجه گیری میکند.

2. کارهای پیشین

نمونه های تجاری پردازنده های شبکه بسیار هستند [3، 6، 7، 10]. یک پردازنده شبکه معمولا به صورت چند پردازنده یک تراشه ای با اجزای I/O باکارایی بالا پیاده سازی میشود که برای پردازش بسته بهینه سازی شده است. به طور خاص، NPها یک معماری مناسب تر نسبت به ایستگاه کاری معمولی یا پردازنده های سرور، برای رسیدگی به این حجم کاری ارائه میکند. نیاز به یک معماری تخصصی به دلیل منحصر به فرد بودن حجم کاریNP است که تحت سلطه بسیاری از تسک های کوچک و عملیات I/O با پهنای باند زیاد است [21]. یک جایگزین دیگر برای پیاده سازی توابع پردازش بسته ، دستگاه های قابل برنامه ریزی منطقی هستند به عنوان مثال، مدارهاس مجتمع دیجیتال قابل برنامه ریزی (FPGA ها)، که بیشتر مناسب برخی از توابع پردازش به سبک جریان داده می باشند [1، 18].

الزامات طراحی برای پشتیبانی زمان اجرا در سیستم های مبتنی بر FPGA بسیار شبیه بهNP های معمولی هستند و در نتیجه انتظار می رود نتایج کار ما نیز به همان اندازه در این دامنه قابل اجرا باشد. به منظور دستیابی به کارایی لازم سرعت خطی روزافزون و توابع پردازش بسته به طور فزاینده پیچیده، هر دو NP و FPGA از روش موازی سازی استفاده میکنند که بطور ذاتی در حجم کاری این سیستم ها وجود دارد. به طور کلی، بسته ها را تا زمانی می توان به صورت موازی پردازش کرد که متعلق به جریان مشابه باشند. توابع پردازشی در یک بسته نیز می توانند بصورت موازی برای کاهش تاخیر بسته اجرا شوند. این امر منجر به سیستمهای NP با پردازش موازی زیاد و موتورهای پردازشی میشود. برای برنامه نویسی چنین سیستمی، چند زبان برنامه نویسی خاص دامنه توسعه یافته است.اینتل به طور مشترک با پروژه Shangri-La در دانشگاه تگزاس در آستین Bakerرا توسعه داده است [5]. پروژه MESCALدر دانشگاه برکلی NP-Click را توسعه داده [17]، که براساس روتر ماژولار Click می باشد[8].

در پروژه NEPAL [12]، Memik و همکاران یک سیستم زمان اجرا ارائه کردند که اجرای برنامه های کاربردی بر روی یک پردازنده شبکه را کنترل می کند. برنامه های کاربردی با استفاده از NEPAL APIبه ماژولهایی در سطح تسک تقسیم شده و که صورت پویا به هسته های اجرا نگاشت میشوند.همچنین یک سطح اضافی ترجمه در طی تبدیل کد برنامه به ماژول وجود دارد.

تجزیه و تحلیل تکنیک های زمان اجرا به ترکیبی از الگوریتم های تخصیص ابتکاری محدود شده است. Teja [19] یک محیط برنامه نویسی تجاری برای خانواده پردازنده های شبکه اینتل IXP می باشد. در حالی که آنها یک سیستم عامل پردازش شبکه ارائه میکنند (NPOS)، هم Tejaو هم NEPAL برای ساده سازی روند برنامه نویسی یک برنامه واحد و با هدف استفاده مجدد کد در تمام سیستم عاملها طراحی شده اند. ما از توانایی انطباق سریع با برنامه های متعدد بر روی همان سیستم پردازنده شبکه پشتیبانی نمی شود.

بااینکه کارهای زیادی بر روی مکانیزم های مدیریت پویای برنامه های متعدد وجود ندارد، پژوهش هایی بر روی الگوریتمهای تطبیق و زمانبندی پردازنده های شبکه برای صرفه جویی در مصرف انرژی انجام گرفته است. Kokku و همکاران مسائل مربوط به طراحی محیط زمان اجرا، شبیه به آنچه که در اینجا ارائه میشود، را بررسی کرده اند[9]،

اما آنها برنامه های پارتیشن بندی شده که روی چند پردازنده شبکه توزیع شده اند را در نظر نگرفته اند. در عوض، فرض شده است که یک برنامه کامل به یک پردازنده تک هسته ای نگاشت میشود.

3. سیستم عامل پردازنده شبکه

اصطلاح "سیستم عامل" اغلب در زمینه ایستگاه های کاری کامپیوتری استفاده می شود. مسئولیت های چنین سیستم عاملی، مدیریت منابع سخت افزاری و جداکردن کاربران از یکدیگر است. هدف بهینه سازی معمولا به حداقل رساندن زمان اجرای یک وظیفه می باشد (وظیفه ای که کاربر فعلا با آن کار میکند). قابل توجه است که اهداف یک سیستم عامل برای پردازنده های شبکه بسیار متفاوت هستند. در یک پردازنده شبکه، تمام برنامه ها توسط یک موجودیت اداری کنترل شده و بهینه سازی با هدف به حداکثر رساندن توان عملیاتی کل سیستم انجام میگیرد.

3.1 تفاوت ها با سیستم عامل های ایستگاه کاری

در زیر جزئیات تفاوت بین سیستم عامل های پردازنده شبکه و سیستم عامل های معمولی لیست شده است:

• تفکیک کنترل و مسیر داده. این جدایی به شرایط پردازنده اشاره دارد نه به شرایط شبکه. برای رسیدن به توان عملیاتی بالا درNP ها، مطالعات متعددی نشان داده اند که پیاده سازی تعداد بیشتری از موتورهای پردازشی ساده تر نسبت به موتورهای کمتر ولی قوی تر مقرون به صرفه تر است این پردازنده های ساده قابلیت اجرای وظایف کنترلی پیچیده در بالای بسته پردازش وظایف ندارد. در طراحی NP های امروزی، کنترل بر روی یک پردازنده کنترل جداگانه پیاده سازی میشود. با توجه به تفکیک بین "کلاس های" پردازنده، داشتن یک ساختار کنترلی واضح تر از یک سیستم عامل معمولی ضروریست.

• تعامل محدود. کاربران به طور مستقیم با NP و یا سیستم عامل آن تعامل ندارند. حداکثر، بعضی اوقات برنامه های کاربردی نصب شده و حذف میشوند. این بدان معنا نیست که یک کاربر نمی تواند تنظیمات یک NP را تغییر دهد

(به عنوان مثال، قوانین به روز رسانی یک نرم افزار دیوار آتش)، اما متغیر کلیدی در این سیستم الگوهای ترافیکی ست که تعیین می کند چه پردازشی باید اتفاق بیافتد.

• نظم و سادگی نرم افزارها. یکی از جنبه های برجسته پردازش شبکه این است که پردازش مسیر داده در بسته های منفرد انجام می گیرد. این بدان معنی است که تسک های پردازش بسته معمولا به علت قیود زمان واقعی تحمیلی توسط بسته های مداوم پیچیدگی ندارند. در نتیجه، تقاضای پردازش کم است (از چند صد تا چند هزار دستورالعمل) [16]. علاوه بر این، مسیر اجرای یک برنامه در بسیاری از موارد مشابه بوده و تنها برای تعداد کمی از موارد دیگر متفاوت است. بنابراین تجزیه و تحلیل جزئی برنامه های کاربردی پردازش بسته برای یافتن نگاشتهای خوب پردازنده امکان پذیر است.

• پردازش غالب بر مدیریت منبع. سیستم عامل های معمولی نیاز به پیاده سازی تعدادی توابع مختلف دارند: زمانبندی پردازنده، مدیریت حافظه، جداسازی نرم افزار، انتزاع منابع سخت افزاری، و غیره. در سیستم های پردازنده شبکه، این چالش ها تحت سلطه مدیریت منابع پردازشی است. تنوع منابع سخت افزاری محدود بوده و بسیاری از آنها به طور مستقیم توسط نرم افزار کنترل می شوند. همچنین، حافظه معمولا بطور ایستا تخصیص داده میشود تا از رفتار قطعی زمان اجرا اطمینان حاصل شود. این مساله ممکن است در آینده تغییر کند همانگونه که برنامه های کاربردی شبکه پیچیده تر شده و سیستم عاملهای پردازنده شبکه بیشتر شبیه به سیستم عامل های معمولی میشوند. در این کار، ما روی جنبه های پردازشی سیستم عامل تمرکز داریم.

• عدم تفکیک فضای کاربر/ فضای کرنل. تمامی توابع در یک پردازنده شبکه توسط یک موجودیت اداری مشابه کنترل می شود. هیچ جدایی واضحی میان فضای کاربر و فضای کرنل وجود ندارد. در عوض، عملیات بین کنترل و مسیر داده تقسیم شده است. در نتیجه مکانیزم های محافظت قدیمی به طور معمول در سیستم های عامل پردازنده شبکه اجرا نمی شوند.

3.2 ملاحظات طراحی

با توجه به تفاوت های زیاد و قابل توجه بین یک سیستم عامل معمولی و یک پردازنده شبکه، به اعتقاد ما کشف برخی از مسائل اساسی مربوط به طراحی که در زمینه سیستم عامل های پردازنده شبکه با آن سروکار داریم، حائز اهمیت است. پرسش هایی که از این اکتشاف بوجود می آیند عبارتند از:

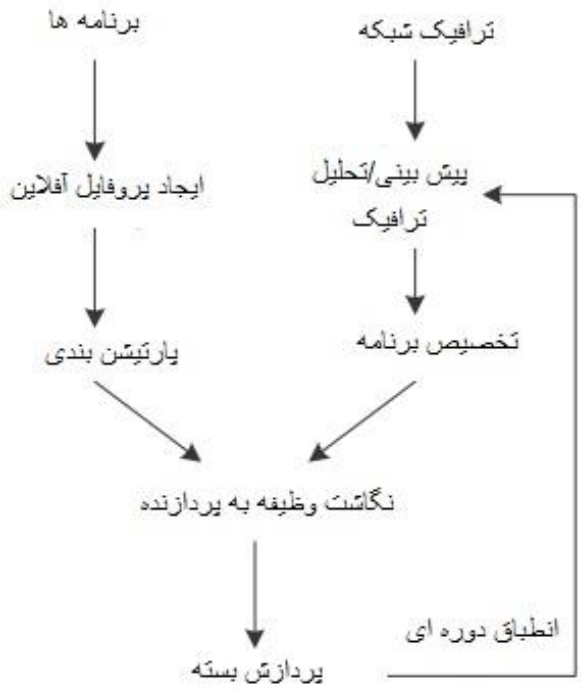
• چگونه برنامه های کاربردی به صورت پویا بر روی پردازنده شبکه نصب و حذف میشوند؟

• چگونه می توان برنامه های کاربردی جهت استفاده از سخت افزار های زیربنایی چندپردازنده را پارتیشن بندی و نگاشت کرد؟

• چگونه باید سیستم عامل را با تغییرات ترافیکی شبکه که نیاز به پیکربندی های مختلفی دارد منطبق نمود؟

• میزان و سطح سازگاری چقدر است؟

به منظور بررسی صحیح جنبه های طراحی سیستم عامل، فرض میکنیم یک رویکرد عملیاتی جامع همانند شکل 1 فرض می کنیم. چهار مرحله اساسی که برای پشتیبانی زمان اجرا سیستم های پردازنده شبکه لازم است وجود دارد: تحلیل نرم افزار ، مشخصات ترافیکی، نگاشت حجم کار، و سازگاری . یک مساله اساسی وجود دارد آنچه که باید بصورت آفلاین انجام شود (به عنوان مثال، در طول توسعه نرم افزار) و آنچه که باید (و می تواند) در طول زمان اجرا انجام شود.



شکل 1 . تحلیل پردازش و ترافیک در یک سیستم عامل پردازنده شبکه

تحلیل نرم افزار برای تجزیه و تحلیل نیازمندی های پردازشی برنامه و پارتیشن بندی نرم افزار ضروریست. پارتیشن بندی اجازه می دهد تا تسکهای فرعی مختلف بر روی عناصر پردازشی مختلف برای استفاده کامل از منابع NP توزیع شوند. سادگی و تکرار برنامه های کاربردی پردازش شبکه [21، 16] اجازه تجزیه و تحلیل دقیقی از نرم افزار را می دهد. فرآیند ایجاد پروفایل به عنوان یک مولفه آفلاین نشان داده شده است. با منابع پردازشی محدود در معماری های فعلی NP، این تجزیه و تحلیل نمی تواند به صورت آنلاین انجام شود. به طور معمول، چنین تحلیلی را می توان در چارچوب محیط توسعه نرم افزار برای برنامه های کاربردی NP اجرا نمود. عمل پارتیشن بندی را می توان به روش های مختلف انجام داد و سطح دانه بندی (granularity) که باید انجام شود با جزئیات بیشتری در زیر مورد بحث قرار گرفته است.

مشخصات ترافیکی شبکه یکی دیگر از جنبه های مهم پشتیبانی زمان اجرا برای سیستم های پردازنده شبکه است. بسته به شرایط ترافیکی فعلی شبکه، برنامه های مختلف بر پردازش مسلط هستند. حجم کار به صورت پویا متغیر دلیل اصلی برای ضرورت پشتیبانی زمان اجرا می باشد. به منظور دستیابی به تخصیص خوب منابع پردازشی، لازم

است بدانیم که چه پردازشی در حال حاضر برای بسته بندی در سیستم ضروری است (و یا در آینده نزدیک پردازش خواهد شد). نتیجه تجزیه و تحلیل ترافیک، تخصیص برنامه است که نسبت پردازش مورد نیاز برای هر نرم افزاری که بر روی سیستم در دسترس است را توصیف می کند.

نگاشت حجم کار فرایندی ست که وظایف پردازشی را به موتورهای پردازش واقعی اختصاص میدهد. این انتساب برمبنای تخصیص و پارتیشن بندی نرم افزار به دست آمده در دو مرحله قبلی می باشد. نگاشت را می توان به راه های مختلف و بسته به معماری خاص سیستم، محیط توسعه نرم افزار و اصول عملیاتی یک سیستم انجام داد. اهداف نگاشت ، رسیدن به توان عملیاتی بالا و بهره برداری کارآمد از منابع می باشد.

مرحله سازگاری نیاز به پیکربندی مجدد سیستم پردازنده شبکه برای مطابقت با نیازهای پردازشی را نشان می دهد که توسط حجم کار ترافیکی دیکته شده است. در طول این مرحله، تخصیص نرم افزار با توجه به شرایط جدید ترافیکی تغییر میکند. سپس، مرحله نگاشت تخصیص وظایف به پردازنده ها را برای مطابقت با تخصیص های جدید اصلاح میکند.

4. مبادلات کیفی

با هریک از چهار مولفه یک سیستم عامل پردازنده شبکه که در بخش قبل مورد بحث قرار گرفت، تعدادی معاوضه کیفی طراحی وجود دارد. ما درباره گزینه های مختلف طراحی در این بخش بحث کرده و پس از آن نتایج کمی را در بخش 5 ارائه میکنیم. از آنجا که نتایج کمی به شدت به یک سیستم خاص وابسته اند، بحث مبادلات را برای حفظ کاربرد عمومی آن جدا کرده ایم.

4.1 پارتیشن بندی نرم افزار

برنامه های کاربردی پردازنده شبکه به ندرت از یک تکه کد یکپارچه تشکیل شده اند. معمولا بیشتر برنامه های کاربردی NP به چند وظیفه فرعی تقسیم می شوند. به عنوان مثال، در اینتل IXP2400، پردازش ورودی از عمل

فوروارد و از پردازش خروجی جدا شده است. این پارتیشن بندی باعث می شود تا توسعه نرم افزار تا حدی آسان تر شده و تغییرات در نرم افزار ساده تر پیاده سازی شود. همچنین بهره برداری از موازی سازی و راه اندازی خط لوله برای استفاده کامل زیرساخت چند پردازنده را اجازه می دهد. اما چگونه یک سیستم عامل می تواند از این پارتیشن بندی پشتیبانی کند؟

4.1.1 پارتیشن بندی دستی

رایج ترین روش برای تفکیک مناسب وظایف و یک نگاشت از وظایف به پردازنده هاست. با استفاده از محیطهای شبیه سازی، برنامه نویسان می توانند پارتیشن بندی خاصی اعمال کرده و نتایج عملکرد را به دست آورند. با اجرای پارتیشن بندی بصورت دستی، نقاط گلوگاهی می توانند حذف شوند و میزان کارایی را می توان بخوبی تنظیم نمود.این روش بسیار زمان بر بوده و نیاز به آگاهی دقیقی از برنامه و سخت افزار NP دارد. از دیدگاه سیستم عاملی، پارتیشن بندی دستی برنامه میزان پشتیبانی پویا را محدود میکند و به طور کلی انطباق با تغییر شرایط ترافیکی میسر نیست.

4.1.2 پارتیشن بندی خودکار

اخیرا، روش های متعددی برای پارتیشن بندی خودکار برنامه های کاربردی ایجاد شده است. جنبه نگاشت اتوماتیک مربوط به این موضوع در زیر مورد بحث قرار گرفته است. یک کامپایلر پارتیشن بندی خودکار توسط اینتل در [5] توسعه داده شده است. Ramaswamy و همکاران یک الگوریتم خوشه بندی آموزش مبتنی بر پروفایل توسعه دادند که ناشی از نمایش یک گراف بدون دور مستقیم از برنامه های کاربردی NP می باشد[15]. دانه بندی از پارتیشن بندی را می توان در صورت نیاز اعمال نمود. Plishker و همکاران یک رویکردی پیشنهاد دادند که در آن برنامه های کاربردی در یک زبان دامنه خاص توصیف شده و سپس بر روی المانهای پردازشی توزیع میشوند [14].

4.1.3 گزینه های طراحی

یک سوال کلیدی این است که کدام سطح دانه بندی از پارتیشن بندی نرم افزار مناسب تر می باشد. طیف گزینه ها از برنامه های کاربردی یکپارچه تا دستورالعمل های تخصیص منابع پردازشی به شدت ریز دانه (یا بلوک پایه) محدود میشوند. مبادلات طراحی عبارتند از:

• نرم افزارهای یکپارچه. اگر نرم افزار پارتیشن بندی نشده باشد، تنها می توان آنرا به یک موتور پردازشی تک اختصاص داد. این امر به طور چشمگیری نحوه انطباق حجم کار برنامه را به نیازمندی های ترافیکی شبکه محدود میکند. همچنین، ممکن است و موجب ایجاد گلوگاه های عملکردی در سیستم های خط لوله شود (به عنوان مثال، چند برنامه متوالی در هر بسته)، که در آن سرعت خط لوله با حداکثر زمان مرحله تعیین میگردد. در نهایت همانطور که اندازه نرم افزار رشد میکند، برنامه های کاربردی یکپارچه اجازه توزیع مقیاس پذیر پردازش را نداشته و ممکن است با محدودیتهای فرمانهای ذخیره سازی تضاد داشته باشند.

• پارتیشن بندی بسیار ریز دانه. حالت مخالف یک نرم افزار یکپارچه پارتیشن بندی شده حالتیست که در آن هر دستورالعمل به صورت جداگانه به یک منبع پردازشی نگاشت شده است. این رویکرد انعطاف پذیری بیشتری فراهم کرده و درضمن سربار بیشتری برای مدیریت نصب و کنترل نرم افزار تولید میکند. همچنین پیچیدگی مساله نگاشت را افزایش میدهد، زیرا تعداد زیادی گره باید نگاشت شوند و فضای راه حل های به طور قابل توجهی با تعدادی از گزینه های نگاشت رشد می کند.

• پارتیشن بندی متعادل. در حالت ایده آل، دوست داریم یک پارتیشن بندی متعادل پیدا کنیم که اجازه توزیع کارآمد وظایف پردازشی را بدهد، اما پیچیدگی مساله نگاشت را کوچک نگه دارد.

در بخش زیر، نشان می دهیم که پارتیشن بندی بهینه و متعادلی وجود دارد که بر اهمیت توجه به پارتیشن بندی نرم افزار برای پردازنده های شبکه تاکید میکند. فرض بر اینست که برنامه های کاربردی دارای یک ساختار یکپارچه اند که در آن نیاز به نگاشت در یک موتور پردازش واحد کافی نیست.

4.2 خصوصیات ترافیکی

خصوصیات ترافیک یک ورودی مهم برای تعیین تخصیص مناسب برنامه های مختلف بر روی پردازنده شبکه است. برنامه های کاربردی به شدت مورد استفاده معمولا برای ارائه کارایی کافی نیاز به چندین تکرار دارند (به عنوان مثال، برنامه های IP فوروارد موازی چندگانه). با درنظر گرفتن پشتیبانی زمان اجرا برای وظایف سنگین، قابلیت تجزیه و تحلیل ترافیک شبکه به منظور برآورد و احتمالا پیش بینی تخصیص نرم افزار حائز اهمیت است.

4.2.1 مدل ترافیکی ایستا

ساده ترین حالت خصوصیات ترافیکی یک مدل ترافیک ایستا می باشد. این مدل بیشتر مورد استفاده است چرا که به تغییر آنلاین در سیستم نیاز ندارد. فرض بر این است که شرایط ترافیکی روی کل زمان اجرای سیستم به صورت یکسان باقی می ماند. تغییرات کوتاه مدت توسط عمل بافرینگ (در نتیجه افزایش تاخیر بسته) و یا عمل بیش تأمین که در آن منابع اضافی به هر نوع نرم افزار اختصاص داده میشود (در نتیجه افزایش کل منابع سخت افزاری مورد نیاز) خنثی می شوند.

4.2.2 مدل دسته ای یا Batch

در روش پردازش دسته ای، تعداد مشخصی از بسته ها بافر شده و نیازمندیهای پردازشی آنها قبل از تخصیص حجم کار مورد تجزیه و تحلیل قرار میگیرد. این عمل بافرینگ خصوصیات دقیق نیازمندی های را نشان میدهد. از آنجا که تخصیص و یا تخصیص مجدد وظایف نمی تواند با سرعت دلخواه انجام شود، حفظ یک نگاشت پردازنده خاص برای مقدار مشخصی از زمان ضروریست. این دوره ممکن است تنها چند میلی ثانیه طول بکشد، اما با توجه به نرخ لینک بالا برابر با حداقل چند ده یا چند صد بسته است. اگر پردازش دسته ای مورد استفاده قرار گیرد، پس از آن تمام بسته ها در یک دسته نیاز به بافرینگ دارند. از آنجا که این امر تاخیر کلی یک بسته را افزایش میدهد، پردازش دسته ای پیش بینی شده ممکن است مناسب تر باشد.

4.2.3 مدل دسته ای پیش بینی شده

در بسیاری از موارد، ترافیک شبکه مقدار مشخصی از لوکالیتی زمانی را نشان میدهد. این می تواند برای برآورد نیاز ترافیکی یک دسته با مشاهده یک پنجره کوچکتر درون آن دسته ("نمونه برداری") مورد استفاده قرار گیرد. براساس مشاهدات، کل نیازهای پردازشی دسته را می توان تعمیم داد. برای نمونه های کوچک، تنها چند بسته برای تعیین تخصیص یک دسته نیاز به بافرینگ دارند. در نمونه های بزرگتر دقت و صحت همراه با تاخیر پردازش افزایش می یابد.

4.2.4 گزینه های طراحی

به منظور تعیین تخصیص برنامه های کاربردی به سیستم پردازنده شبکه، خصوصیات ترافیک را می توان با توجه به روش های فوق اعمال نمود. تغییرات عبارتند از:

• انتخاب اندازه دسته. اندازه دسته تعیین میکند که چند بسته با همان تخصیص منابع پردازشی به برنامه، پردازش میشوند. هرچه اندازه دسته کوچکتر باشد، این تخصیص را می توان بیشتر با تغییرات ترافیکی منطبق نمود. با این حال دسته های کوچکتر، زمان کمتری بین پیکربندی دوباره سیستم عامل برای تعیین نحوه انطباق و یافتن یک نگاشت وظیفه خوب صرف میکنند.

• انتخاب اندازه نمونه. به منظور به حداقل رساندن تاخیر بسته ها، نمونه باید کوچک انتخاب شود. با اینکار دقت کاهش یافته و در نتیجه محدودیتی در انتخاب اندازه کوچک بودن آن وجود دارد.

• دانه بندی تخصیص برنامه. از میان مدل های ترافیکی، یک تخصیص برنامه مشتق میشود که بخشی از نیازهای پردازشی کلی را به هر نرم افزار واگذار میکند. برای دسته ها و نمونه های بزرگ، این تخصیص را می توان با دقت بالا مشخص نمود. در مقابل، بسیاری از سیستم های پردازنده شبکه تنها از تخصیص دانه درشت برنامه های کاربردی پشتیبانی می کنند (به عنوان مثال، تعداد دفعاتی که یک برنامه خاص بر روی چندپردازنده تکرار میشود). بنابراین،

دانه بندی تخصیص برنامه را می توان در همان سطح از دقت نگه داشت که پردازنده شبکه می تواند از آن پشتیبانی کند. این امر استفاده از نمونه های کوچک تر را مجاز میدارد.

4.3 نگاشت و تطبیق زمان اجرا

نگاشت وظایف نرم افزار به المانهای / پردازشی از طریق یک الگوریتم نگاشت انجام میگیرد. ما مسائل طراحی را بدون نیاز به استفاده از یک الگوریتم خاص، بررسی میکنیم. با این حال، فرض می کنیم دو ویژگی در الگوریتم های نگاشت وجود دارد:

• الگوریتم نگاشت می تواند بطور تدریجی به نتایج بهتری منجر شود، همانطور که زمان اجرا افزایش می یابد. این بدان معنی ست که طراح سیستم عامل می تواند زمان اجرای الگوریتم و کیفیت از نگاشت حاصل را انتخاب نماید.

• الگوریتم نگاشت را می توان در یک سیستم تا حدی پیکربندی شده اعمال کرد. این یعنی برخی از برنامه ها ممکن است حذف شده و الباقی را می توان اضافه نمود، بدون تغییر نگاشت برنامه هایی که تحت تاثیر قرار نگرفته اند.

گزینه های طراحی حاصل به میزان و سطح نگاشت (جزئی) رسیدگی میکنند.

4.3.1 نگاشت ایستا

نگاشت ایستا دست به دست همراه با پارتیشن بندی ایستا و یک مدل ترافیکی ایستا می باشد. در این مورد، وظایف پردازشی به طور آفلاین به پردازنده ها اختصاص داده شده و هیچ تغییری در طول زمان اجرا صورت نمیگیرد.

4.3.2 نگاشت پویای کامل

نگاشت کامل به یک راه حل نگاشت اشاره دارد که در آن کل حجم کار از ابتدا نگاشت شده است. این الگوریتم می توانید وظایف پردازشی را بر روی کل معماری بدون هیچ گونه محدودیت اولیه قرار دهد. و به طور معمول به یک راه حل خوبی منجر میشود که نقطه بهینه نظری برای زمان پردازش درحال افزایش را تخمین میزند.

4.3.3 نگاشت پویای جزئی

نگاشت جزئی فرض می کند که بخشی از حجم کار در حال حاضر روی سیستم NP نگاشت شده است. این الگوریتم تنها نیاز به نگاشت چند برنامه کاربردی به منابع پردازش باقی مانده دارد. این رویکرد نسبت به نگاشت پویای کامل محدود تر است، زیرا اکثر منابع سیستم در حال حاضر به یک برنامه اختصاص داده شده اند. ماهیت رو به رشد این روش، با ریسکی روبروست که الگوریتم نگاشن در یک نقطه حداقل محلی "گیر می کند". با این وجود، هزینه های پردازش برای نگاشت جزئی کمتر از نگاشت کل حجم کار می باشد.

4.3.4 تلاش های نگاشت

یک پارامتر جالب دیگر برای نگاشت پویا این است که هرچند وقت یکبار سیستم را مجددا برنامه ریزی کنیم. در حالت ایده آل، می خواهیم تخصیص نرم افزار را با هر بسته برای تضمین بهترین کارایی سیستم و عملکرد بالا پیکربندی مجدد نمائیم، در حالی که ترافیک تغییر میکند. با این حال، هزینه های مرتبط با نگاشت و نگاشت مجدد وجود دارد. به غیر از هزینه آپلود دستورالعملهای جدید در هر پردازنده، تعیین نگاشت جدید به قدرت پردازشی و زمان محاسبه نیاز دارد. نگه داشتن فرکانس برنامه ریزی مجدد در یک نرخ پایین حائز اهمیت است، که در آن زمان پردازش کافی برای پیدا کردن نتایج نگاشت در دسترس باشد.

4.3.5 گزینه های طراحی

گزینه های طراحی برای نگاشت تعیین میکند که هرچند وقت یکبار، چقدر و چه میزان تلاش برای انجام نگاشت کامل و یا جزئی مورد نیاز است.

• گزینه میزان انطباق پویا و تلاش نگاشت. به منظور انطباق با تغییر شرایط ترافیکی، شبکه سیستم عامل پردازنده نیاز به تغییر تخصیص برنامه و در نتیجه عمل نگاشت دارد. سرعتی که در آن این مساله اتفاق می افتد حداکثر زمان در دسترس برای تعیین نگاشت جدید را مشخص می کند. هرچه سرعت انطباق کمتر باشد، زمان بیشتری برای

یافتن راه حل نگاشت بهتر صرف میشود. همانطور که نرخ تطبیق افزایش می یابد، کیفیت راه حل نگاشت مشتق شده نیز کاهش می یابد.

• گزینه نگاشت جزئی و تلاش نگاشت. تغییرات در شرایط ترافیکی ممکن است تنها روی چند برنامه تاثیر بگذارد. به منظور قابلیت انطباق سریع با هزینه نگاشت کمتر، یک طراح سیستم عامل ممکن است تنها بخش کوچکی از تخصیص را برای نگاشت انتخاب کند. مزیت این توانایی، انطباق سریع است اما میزان تغییرات ترافیکی که میتوان پشتیبانی شود محدود به بخشی از NP است که دوباره نگاشت شده است.

• تعادل نگاشت جزئی و کامل. نگاشت جزئی تکراری باعث میشود تا راه حل های نگاشت بدتر شوند. به منظور جلوگیری از این مساله، مراحل نگاشت کامل باید به صورت دوره اس انجام شود. هرچقدر این اتفاق بیشتر می افتد، کمتر احتمال دارد که سیستم به یک وضعیت ناپایدار برود. با این حال، این امر تلاش های نگاشت کلی را افزایش می دهد.

نتایج کمی نشان می دهد که نگاشت جزئی می تواند سازگاری سریعتری به تغییرات ترافیکی نشان دهد، اما بدتر شدن نتیجه نگاشت می تواند تاثیر قابل توجهی روی کارایی داشته باشد.

4.4 محدودیت ها

یک پردازنده شبکه تعدادی محدودیت سیستمی دارد که در موارد فوق در نظر گرفته نشده است. این محدودیت ها می تواند نقش مهمی در هنگام تصمیم گیری طراحی بازی کند.

4.4.1 محدودیت های ذخیره سازی دستورالعمل

اکثر سیستم های پردازنده شبکه به شدت در میزان ذخیره سازی دستورالعمل که برای هر موتور پردازش در دسترس است محدود شده اند. این به خاطر هزینه نسبتا بالای سطح تراشه حافظه در مقایسه با منطق پردازش است. این محدودیت به این دلیل است که تمام برنامه های کاربردی را می توان روی تمام موتورهای پردازش در تمام زمان

ها نصب نمود. بنابراین تغییرات نگاشت بسیار پر هزینه اند ازآنجایی که به آپلود دستورالعمل جدید به هر پردازنده نیاز است.

4.4.2 بیش تامین

یک سیستم پردازش شبکه باید قادر به پردازش هر بسته ای که بر روی شبکه منتقل می شود باشد. از سوی سیستم عامل پردازنده شبکه، این بدان معنی است که منابع پردازشی باید برای تمام برنامه ها در دسترس باشند. اگر این مورد نباشد، بسته ها باید تا چرخه تخصیص بعدی صبر کرده و یا در مسیر آهسته روتر پردازش شوند. یکی از راه های جلوگیری از این تاخیر ، بیش تامین سیستم و افزایش تخصیص نرم افزار به بیش از 100٪ است. می توان تضمین کرد که حتی برنامه های کاربردی که انتظار ترافیک در دسته های آینده ندارند، می توانند فقط در این مورد نصب شوند.

5. نتایج کمی

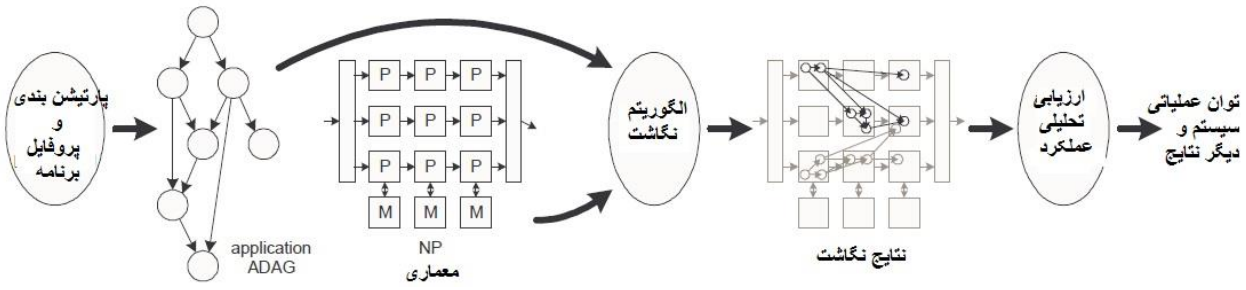
در این بخش، مشاهدات کیفی بخش های قبل با نتایج کمی بررسی میشوند. این نشان میدهد که کدام روند تاثیر زیاد و کدام مساله تاثیر کمتری بر روی عملکرد سیستم داشته است.

به منظور استخراج نتایج کمی، از یک سیستم مبنای خاص استفاده کرده ایم. البته، تفاوت های زیادی بین سیستم های مختلف پردازنده شبکه و نتایج بر روی یک سیستم دیگر وجود دارد. از این رو در نظر گرفتن روندهایی که می تواند در نتایج ما مشاهده شود (مثلا آیا یک نقطه بهینه وجود دارد؟) نسبت به نقاط داده منفرد (به عنوان مثال محل دقیق نقطه بهینه ؟) حائز اهمیت می باشد.

5.1 سیستم مبنا

معیاری که بدان علاقه مند هستیم توان عملیاتی یک سیستم پردازنده شبکه با توجه به یک حجم کار خاص است. به منظور استخراج این اطلاعات، نیاز به اجرای برخی از توابع داریم که برای سیستم عامل های شبکه ضروری اند. مخصوصا نیاز داریم برنامه های کاربردی واقعی پردازش شبکه، پارتیشن بندی و نگاشت وظایف پردازش در موتورهای پردازشی را در نظر بگیریم. به منظور بررسی فضای طراحی که توصیف کرده ایم، تنها در نظر گرفتن تعداد انگشت شماری از راه حل های پارتیشن بندی و نگاشت کافی نیست. بنابراین استفاده از یک روش مدل سازی تحلیلی به جای شبیه سازی را انتخاب کردیم. با پارتیشن بندی خودکار، نگاشت و محیط مدل سازی ارائه شده در [15] و [20]، می توانیم تعداد زیادی از پارتیشن بندی های ممکن، نتایج نقشه برداری و غیره را ارزیابی کنیم. این امر یک درک اولیه از مسائل کمی فراهم می کند. در این روند، چند معیار فرعی (به عنوان مثال، هزینه استخراج یک نگاشت با کیفیت خاص) را می توان به دست آورد.

روند به دست آوردن نتایج عملکرد در شکل 2نشان داده شده است. به طور خلاصه سه مولفه کلیدی پارتیشن بندی و نمایش نرم افزار، الگوریتم های نگاشت و مدل عملکرد تحلیلی را برای ارائه مبنایی برای درک نتایج زیر توصیف می کنیم.



شکل 2. تحلیل نرم افزار، نگاشت و فرایند ارزیابی کارایی. معمولا ADAG های چندگانه در معماریNP برای منعکس کردن حجم کار نگاشت میشوند که میتوانند توسط پردازنده شبکه پردازش شوند.

5.1.1 نمایش نرم افزار

یک نرم افزار پردازش شبکه باید در راهی نشان داده شود که بتوان آنرا به راحتی به چند المان پردازش موازی و یا خط لوله نگاشت کرد. این به یک نمایشی نیاز دارد که موازی سازی برنامه را به نمایش بگذارد، در عین حالی که اطمینان حاصل کند که وابستگی داده و کنترل در نظر گرفته شده باشد. ما از یک گراف بدون حلقه جهتدار حاشیه دار (ADAG) برای نمایش مشخصات اجرای پویای برنامه های کاربردی استفاده میکنیم.

این ADAG از پروفایل پویای نرم افزار با تعیین وابستگی داده و کنترل بین دستورالعمل های منفرد مشتق میشود. با استفاده از یک روش ابتکاری خوشه بندی که سربار ارتباطی را به حداقل می رساند، دستورالعمل ها در گره های بزرگتر گراف جمع میشوند. هر گره با اطلاعاتی در مورد تعداد کل دستورالعمل ها و دسترسی به حافظه که هنگام پردازش گره اجرا می شود، تفسیر میشود.

5.1.2 الگوریتم نگاشت

هنگامی که برنامه را بصورت یک ADAG نمایش دهیم، گام بعدی نگاشت این ADAG روی یک توپولوژی NP است. هدف از این نگاشت تخصیص وظایف پردازش (به عنوان مثال، گره های ADAG) به المانهای پردازشی و تولید یک زمانبندی ست که به حداکثر توان عملیاتی سیستم دست یابد. این تخصیص آسان نیست، زیرا فرایند نگاشت نیاز به در نظر گرفتن وابستگی های درون یک ADAG دارد و باید اطمینان حاصل شود که پردازش صحیح بسته ها امکان پذیر است. علاوه بر این، Malloy و همکاران نشان داده اند که تولید یک زمانبند بهینه برای یک سیستم که شامل هزینه های اجرایی و ارتباطی باشد یک مساله NP کامل است، حتی اگر تنها دو المان پردازشی وجود داشته باشد [11]. بنابراین ما نیاز به توسعه یک روش هیوریستیک برای پیدا کردن یک راه حل تقریبی داریم.

راه حل ابتکاری ما در حل مشکل نگاشت مبتنی بر "نگاشت تصادفی" می باشد. ایده اصلی انتخاب تصادفی یک نگاشت معتبر و ارزیابی عملکرد آن است. با تکرار این روند به دفعات زیاد و انتخاب بهترین راه حل که بیش از همه تکرار شده است، رسیدن به یک تقریب خوب در حد نقطه بهینه جهانی امکان پذیر می باشد. با رویکرد تصادفی هر

راه حل ممکن در نظر گرفته شده و با یک احتمال کم اما غیر صفر انتخاب میشود. این تکنیک پیشنهاد شده و با موفقیت در حوزه های کاربردی مختلف استفاده می شود [13]. عمل نگاشت برای ADAG های متعدد و احتمالا متفاوت انجام شده است. ترکیبی از ADAG ها نشان دهنده تخصیص برنامه های کاربردی به معماری NP می باشد.

5.1.3 مدل عملکرد تحلیلی

به منظور ارزیابی کارایی توان عملیاتی یک راه حل داده شده، ما از یک مدل عملکرد تحلیلی که پردازش، ارتباطات بین پردازنده، رقابت های حافظه و اثرات همگامی خط لوله را در نظر میگیرد، استفاده می کنیم. پس از نگاشت ADAGهای نرم افزار در توپولوژی پردازنده شبکه ، دقیقا حجم کار برای هر المان پردازشی را می دانیم. این اطلاعات شامل تعداد کل دستورات اجرایی، تعداد دسترسی ها به حافظه و میزان ارتباط بین مراحل است. این مدل باید رقابت منابع به اشتراک گذاشته شده (کانال های حافظه و اتصالات ارتباطی) را به حساب آورد. ما به خصوص به حداکثر زمان تاخیر هر مرحله پردازش خط لوله علاقه مند هستیم چرا که سرعت کلی سیستم از این راه تعیین میشود. تعداد ADAGها که به یک معماری نگاشت شده اند، تعیین میکند چند بسته در طول یک مرحله زمانی پردازش شده است. پس از مشخص کردن چند پارامتر سیستمی، توان عملیاتی سیستم برای یک نگاشت می تواند بیان شود.

5.1.4 پیکربندی سیستم و حجم کار

معماری سیستم در مدل فوق را می توان برای نمایش هر معماری NP معمولی با تعداد مشخصی از موتورهای پردازش موازی (PE) و مراحل خط لوله پیکربندی نمود. ما یک سیستم پایه با یک پیکربندی ثابت برای کاوش مسائل سیستم عامل انتخاب کرده ایم (جدول 1). یک پرسش بوجود آمده این است که چگونه این تغییر برای پیکربندی معماری های مختلف به کار می آید. این اکتشافات فضایی طراحی در حال حاضر به کار ما رسیدگی

نمیکند. برنامه های کاربردی که برای این سیستم در نظر گرفته می شوند، طبقه بندی جریان IP و مبتنی بر هش و IP-lookupمبتنی بر درخت radix است.

پارامترهای سیستمی	پیکربندی مبنا
مراحل خط لوله NP	4
به ازای هر مرحله PEs	4
تعداد کل PEs	16
رابط های حافظه در هر مرحله	2
تعداد دفعات دسترسی به حافظه (چرخه)	10
تعداد ADAGs	20
تعداد گره به ازای ADAG	8

جدول 1. پیکربندی مبنا برای نتایج کمی

5.2 نگاشت وظایف پردازشی

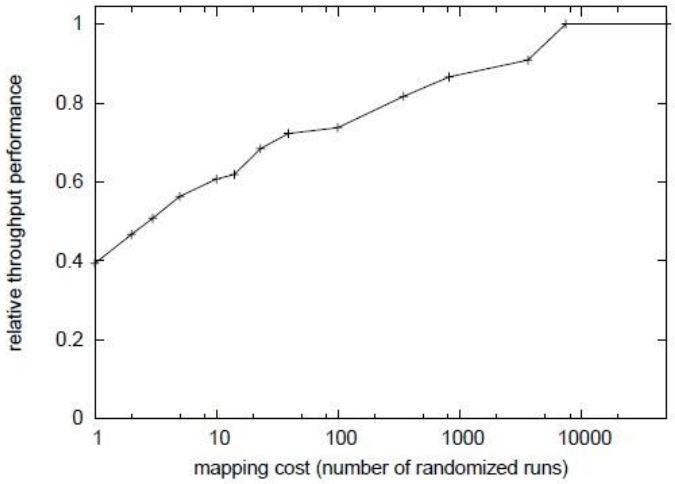
5.2.1 معیارها

نگاشت مقدار مشخصی از پردازش را در بر میگیرد. این به صورت c بیان شده است. کارایی که توسط نگاشت به دست می آید به عنوان t یعنی توان عملیاتی سیستم بیان شده است. با توجه به NP-کامل بودن مساله نگاشت، پیدا کردن یک راه حل کلی بهینه تقریبا غیرممکن است. آنچه که واقعا مطلوب یک سیستم است تعیین یک راه حل "به اندازه کافی خوب" با اجرای یک الگوریتم تخمین برای مدت زمان زیاد می باشد.

5.2.2 نتایج

شکل 3 افزایش کیفیت حاصل از نگاشت به عنوان تلاش پردازشی اختصاص یافته به فرایند نگاشت را نشان می دهد. هزینه نگاشت مدت زمانی است که برای محاسبه راه حل طول می کشد (و به صورت تعداد تکرارهای نگاشت تصادفی بیان میشود). ما سربار اضافی برای متوقف کردن، برنامه ریزی مجدد و راه اندازی مجدد موتور پردازش که

در سیستم های واقعی رخ می دهد را در نظر نگرفته ایم. از آنجا که حداکثر کارایی سیستم مشخص نیست، از یک نتیجه نگاشت با هزینه بسیار بالا به عنوان مبنا استفاده میکنیم.



شکل 3. کارایی سیستم در مقایسه با هزینه عمل نگاشت

5.3 پارتیشن بندی

هدف از پارتیشن بندی نرم افزار مطالعه تاثیر دانه بندی پارتیشن بر روی کارایی است.

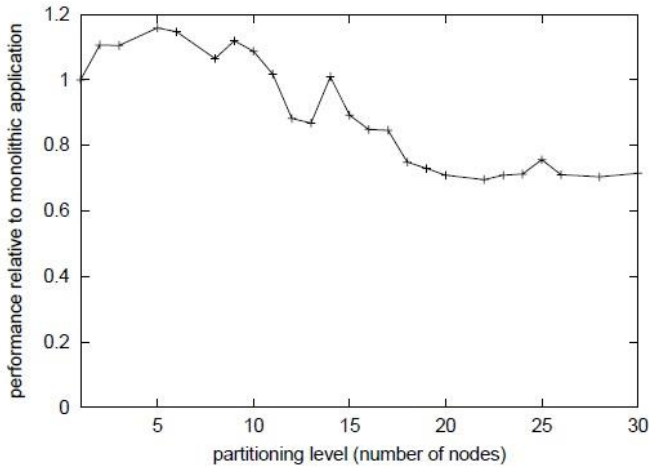
5.3.1 معیارها

تعدادی از وظایف n (یا گره های ADAG) را که در آن برنامه ها پارتیشن بندی می شوند را در نظر بگیرید. مقدار حداکثر n برای هر نرم افزار متفاوت است، اما برای این ارزیابی ما فقط مقادیر n که بسیار کوچکتر از این حداکثر هستند را در نظر گرفته ایم. اگر پارتیشن بندی متعادل باشد، اندازه هر تسک فرعی تقریبا $\frac{1}{n}$ اندازه نرم افزار است.

5.3.2 نتایج

ابتدا مبادلات بین توان عملیاتی سیستم و دانه بندی پارتیشن را کاوش میکنیم. دانه بندی ریز (Fine granularity) منجر به عملکرد بهتر میشود، اگر عمل نگاشت قابل قبول و نامحدود باشد. با توجه به واقعیت های

یک سیستم عامل پردازنده شبکه، اعمال نگاشت به اندازه دسته و میزان انطباق محدود شده است. شکل 4 توان عملیاتی سیستم مبنا را با سطوح مختلف دانه بندی نرم افزار نسبت به پیاده سازی یکپارچه نشان میدهد، که در آن کل نرم افزار بر روی یک پردازنده تک اجرا میگردد. این عمل نگاشت ثابت است و می توان مشاهده کرد که بهترین عملکرد به n=5 رسیده است. نرم افزارهای یکپارچه بدتر اجرا میشوند، چرا که اجازه توزیع وظایف پردازشی بر روی سیستم چند پردازنده را نمی دهد. مقادیر بزرگتر n نیز منجر به کاهش کارایی میشوند، به دلیل آنکه مشکل نگاشت پیچیده تر شده و پیدا کردن یک نگاشت خوب در یک مدت زمان محدود سخت تر خواهد بود.



شکل 4. کارایی سطوح مختلف پارتیشن بندی نرم افزار. نگاشت کل به حدود 10000 گره نگاشت محدود شده است.

5.4 خصوصیات ترافیکی

برای نشان دادن نیاز توصیف ترافیک و سازگاری زمان اجرا، آزمایشات اندازه گیری شبکه انجام شده است.

5.4.1 معیارها

همانطور که در بالا توضیح داده شد، مدل دسته ای قابل پیش بینی فرض می کند که ترافیک در دسته ها (با اندازه دسته b) پردازش شده و تخصیص نرم افزار بر اساس یک نمونه (با اندازه l) است. پس می توانیم تغییرات ترافیکی v

را بر اساس دو معیار$e_{i,j}(a)$ و $p_{i,j}(a)$ توصیف کنیم. e نشان دهنده تعداد تقریبی بسته های موردنیاز نرم افزار a و p تعداد واقعی بسته های موردنیاز نرم افزار در بازه [i...j] است:

$$v_i(l,b) = \frac{1}{b} \cdot \sum_a \max(p_{i,i+b}(a) - \frac{b}{l} e_{i,i+l}(a), 0). \qquad (1)$$

برای مثال، اگر ترافیک دقیقا منطبق با تخصیص تخمین زده باشد، سپس تمام بسته ها "همسان" بوده و تغییر ترافیک V = 0 است. اگر نصف بسته های یک دسته متفاوت با آنچه انتظار می رود باشد (به عنوان مثال، تمام بسته ها به یک نرم افزار واحد نیاز دارند به جای یک برآورد 50/50 بین دو نرم افزار)، تغییرات ترافیک v =0.5 می باشد.

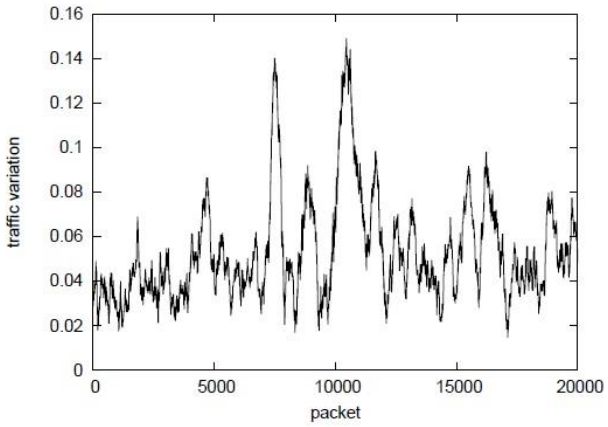
5.4.2 نتایج

برای دریافت داده های واقعی در آنچه تغییرات ترافیکی در یک شبکه به نظر می رسد، ما داده های اندازه گیری را از لینک دسترسی به اینترنت Gigabit خود تعیین می کنیم. حدود 4،235،403 بسته جمع آوری شده و با برنامه های لایه 7 طبقه بندی شده اند (با استفاده از قوانین طبقه بندی ابزار Ethereal). در مجموع 175 طبقه وجود دارد، اما بیش از 98 درصد از ترافیک در پنج دسته قرار گرفته است.

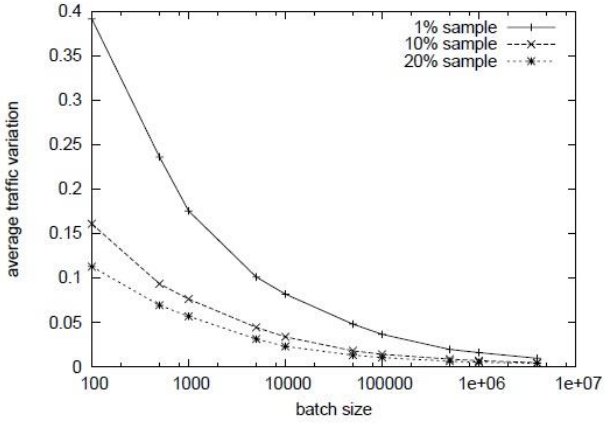
شکل 5 تغییرات ترافیک را روی دنباله ای از بسته ها نشان می دهد. پارامترها b=10,000 و l=100 بوده و تغییرات به صورت یک پنجره کشویی محاسبه می شود. در حالی که این تنها یک نمونه کوچکی از اندازه گیری کلی که انجام داده ایم است، به درستی روند کلی را منعکس میکند. در اغلب موارد، تغییرات در حدود v=0.04 با چند خوشه تا v=0.15 است.

البته، تغییرات مشاهده شده بستگی به کیفیت برآورد (مثلا، اندازه l نسبت به b) و همچنین اندازه دسته b دارد. برای نشان دادن این رابطه، شکل 6 میانگین تغییرات مشاهده شده در کل طرح برای اندازه های مختلف دسته و درصد نمونه های مختلف l/b را نشان می دهد. در نمونه های کوچک و اندازه های دسته ای کوچک میانگین تغییرات بسیار بالا می باشد (V = 0.4). اوج تغییر در این مورد می تواند به v=1برسد.هرچه درصد نمونه بیشتر باشد، بهتر تخمین زده شده و در نتیجه تغییرات ترافیکی کمتر میشود. همانطور که اندازه دسته افزایش می یابد،

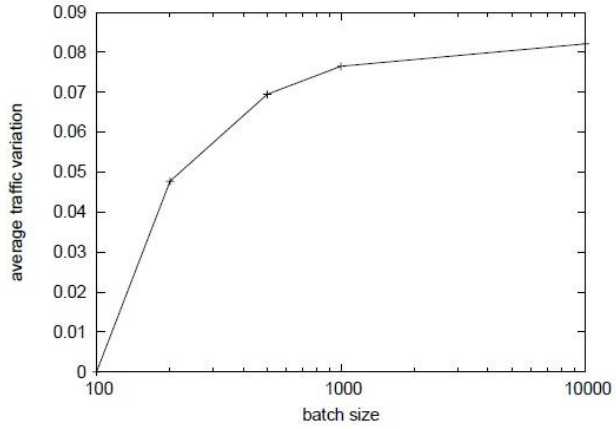
تغییرات موقت درون ترافیک شبکه "هموارتر" شده و متوسط تغییرات کمتری مشاهده خواهد شد. این بدان معنی نیست که رویکرد تخصیص ایستا ($b=\infty$) لزوما ایده آل است. آنچه که این شکل نشان نمی دهد تاخیری خواهد بود که توسط تغییرات ترافیکی با مقیاس کوچکتر و روش بیش تأمین که برای حمایت از تمام شرایط ترافیکی لازم است، ایجاد می شود. هنگامی که مقدار نمونه برداری ثابت است، اندازه دسته بزرگتر منجر به تغییرات ترافیکی بیشتری خواهد شد، همانطور که در شکل 7 نشان داده شده است. این نتایج اهمیت سازگاری زمان اجرا را نشان میدهد به دلیل اینکه ترافیک بطور قابل توجه در مقیاس های زمانی کوتاه تغییر نمیکند.



شکل 5. تغییرات ترافیکی روی ترتیبی از بسته ها



شکل 6. تغییرات ترافیکی در مقایسه با اندازه دسته با سطوح مختلفی از نمونه برداری. اندازه نمونه درصدی از اندازه دسته تنظیم شده است.



شکل 7. تغییرات ترافیکی در مقایسه با اندازه دسته و اندازه نمونه ثابت. اندازه نمونه**l=100** تنظیم شده است.

5.5 انطباق

انطباق در طول زمان اجرا به وسیله تغییرات ترافیکی هدایت می شود. نگاشت کامل و جزئی در طول هر فرایند انطباق انجام میگیرد.

5.5.1 معیارها

معیارهای که برای انطباق مفید هستند، توان عملیاتی سیستم در مقایسه با یک پیکربندی پایه است. پارامترهای ورودی کلیدی میزان انطباق و حجم نگاشت جزئی است (به عنوان مثال، بخشی از منابع NP که می تواند به برنامه های جدید اختصاص یابد). در آزمایش ما، فرکانس انطباق به همان اندازه دسته را در نظر گرفته ایم.

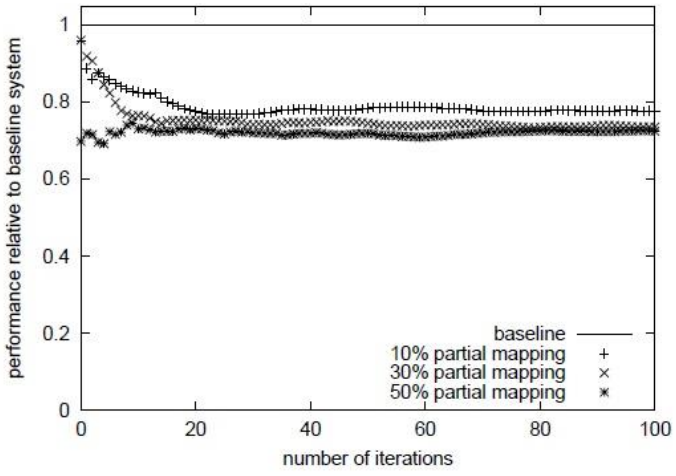
5.5.2 نتایج

شکل 8 اثر کاهش نگاشت جزئی تکرارشونده را نشان می دهد. این شکل نشان می دهد که حذف و اضافات مکرر برنامه های کاربردی باعث میشود عملکرد سیستم به سرعت افت کرده و سپس در یک وضعیت کمتر از حد مطلوب حدود 80 درصد از اوج عملکرد ثابت بماند. تغییرات کمی بین سطوح مختلف انطباق نسبی وجود دارد. ارزش تثبیت

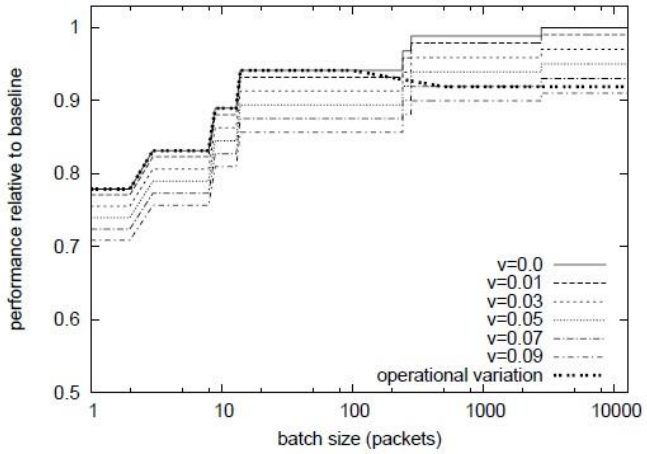
توسط میزان تلاشی که به نگاشت جزئی اختصاص داده شده هدایت می شود. این نتیجه به وضوح نشان می دهد که نگاشت جزئی به سرعت می تواند به پیکربندی بهینه منجر شود. هنگام طراحی یک سیستم عامل NP، باید در نظر داشت که گاهی اوقات نگاشت کامل ضروری است.

فرکانس انطباق (به عنوان مثال، اندازه دسته) با یک معاوضه بین انطباق پذیری سیستم برای تغییر ترافیک و کیفیت نتیجه نگاشت که می تواند مشتق شود روبروست. شکل 9 عملکرد نسبی پیکربندی با اندازه دسته مختلف و تغییرات ترافیکی را نشان می دهد. با افزایش اندازه دسته، کارایی سیستم افزایش می یابد، اما هرچه تغییر ترافیک بیشتر شود، کارایی کاهش می یابد.

خط مشخص شده با عنوان "تغییر عملیاتی" عملکرد سیستم حداکثر با تغییرات واقعی ترافیک و اندازه دسته انتخاب شده را نشان می دهد (با اندازه نمونه ثابت 100=امانطور که در شکل 7 نشان داده شده است). هرچه اندازه دسته افزایش یابد، تغییرات ترافیکی نیز بیشتر میشود. انحراف ناشی از تخصیص پردازش باعث افت کارایی می شود. روند این منحنی نشان می دهد که به وضوح یک اندازه دسته بهینه وجود دارد. این بیشتر از این استدلال حمایت میکند که ملاحظات طراحی خوب در هنگام انتخاب پارامترهای سیستمی بسیار ضروری است.



شکل 8. افت کارایی به علت نگاشت جزئی تکرارشونده. مورد مبنا یک نگاشت کامل است.



شکل 9. کارایی برای اندازه دستهمختلف تحت تغییرات ترافیکی. نگاشت کل ثابت بوده و مورد مبنا دارای اندازه دسته نامحدود بدون تغییرات ترافیکی است.

6. سناریوهای طراحی سیستم عاملNP

در دو بخش قبلی، ما در مورد تعدادی از ملاحظات طراحی بحث کرده و وابستگی کمی و کیفی آنها را کشف نمودیم. برای خلاصه بندی برخی از این مشاهدات و شناسایی نتایج دقیق، سه طرح نمونه برای سیستم عامل های پردازنده شبکه ارائه میکنیم. برای هر سناریو، در مورد جنبه های عملکردی بر اساس مشاهدات فوق بحث خواهیم نمود.

6.1 سناریو 1: پیکربندی ایستا

این سناریو فرض می کند که تمام تجزیه و تحلیل ها، نگاشت ها و عملیات تخصیص بصورت آفلاین انجام میشود. هنگامی که پردازنده شبکه پیکربندی شود، هیچ سازگاری انجام نشده است. این سناریو بسیاری از سیستم های پردازنده شبکه امروزی که برای یک حالت برنامه واحد برنامه ریزی و بهینه سازی شده اند را نشان میدهد. ملاحظات طراحی و عملکردی عبارتند از:

• سادگی سیستم. واضح است، همانند اداره دستی تمام مسائل نگاشت و تخصیص پیاده سازی سیستم ساده میشود. هیچ نیازی به کنترل زمان اجرا وجود ندارد.

• انعطاف پذیری محدود زمان اجرا. روش استاتیک اجازه هیچ گونه تغییری در طول زمان اجرا نمی دهد. هر تغییر در پیکربندی حجم کار نیاز به استفاده از یک ابزار توسعه نرم افزاری برای به پیکربندی مجدد کل سیستم دارد. همانگونه که پردازنده های شبکه به اجزای جدایی ناپذیر شبکه تبدیل شده اند، این رویکرد کمتر عملی خواهد بود.

• افت کارایی تحت تغییر ترافیکی. از شکل 5، می توانید ببینید که شرایط ترافیکی در طول زمان تغییر میکند. این امر منجر به افت عملکرد و یا نیاز به عمل بیش تامین قابل توجه میگردد.

6.2 سناریو 2: تنظیمات از پیش تعیین شده

در این سناریو، فرض می کنیم که سیستم NP را می توان به یکی از چند پیکربندی حجم کار از پیش تعیین شده تنظیم نمود. هر یک از تنظیمات به صورت استاتیک به یک فرایند آفلاین نگاشت میشود. در طول زمان اجرا، سیستم می تواند با هر یک از این تنظیمات منطبق شود. ملاحظات طراحی و عملکردی عبارتند از:

• نگاشت آفلاین. در حالی که سیستم نیاز به نظارت بر تغییرات ترافیکی دارد، نیازی به انجام محاسبات نگاشت نیست. در نتیجه پیچیدگی سیستم عامل را محدود می کند.

• انطباق پذیری محدود. تعداد پیکربندی هایی که می تواند از پیش تعیین گردند، محدود است. اگر ترافیک در خارج از مرزهای برآورد شده تغییر کند، انطباق پذیری امکان پذیر نمی باشد. در چارچوب ترافیک برآورد شده، بعید است که ترافیک واقعی به طور کامل بر یک پیکربندی از پیش تعیین شده منطبق باشد. بنابراین، سطح معینی از افت کارایی را می توان انتظار داشت.

• نتایج نگاشت با کیفیت بهتر. با توجه به در دسترس بودن میزان مطلوبی از قدرت محاسباتی برای نگاشت آفلاین، کیفیت نتایج نگاشت (شکل 3) می تواند بهتر از نگاشت آنلاین باشد.

6.3 سناریو 3: پیکربندی پویای کامل

در سناریوی کاملا پویا، نگاشت بطور آنلاین انجام شده و انطباق سیستم برای مطابقت با تغییرات ترافیکی اجرا میشود. این مورد از روش نمونه گیری به منظور برآورد نیازمندی های پردازشی برای هر دسته استفاده میکند. ملاحظات طراحی و عملکردی عبارتند از:

• انطباق پذیری کامل. یک سیستم کاملا پویا می تواند ب هر تغییرات ترافیکی سازگاری داشته باشد - حتی تنظیماتی که در هنگام برنامه نویسی سیستم پیش بینی نمی شود. این مساله به وضوح مهمترین فایده کاربردی این سناریو است.

• کیفیت نگاشت محدود. با توجه به ماهیت آنلاین فرآیند نگاشت، تنها زمان اجرای محدودی موجود است (اندازه دسته بسیار بزرگ کارایی را کاهش میدهد-شکل 9 را ببینید). بنابراین، کیفیت نتایج نگاشت کمتر از دو سناریوی دیگر است.

• سربار Overprovisioning کمتر. به علت توانایی انطباق با تغییرات شرایط ترافیکی، یک سناریوی به طور کامل پویا می تواند توان عملکردی بالا با منابع پردازشی کمتر فراهم کند.

به طور خلاصه ، هر یک از این سه روش دارای مزایای خاص خود اند (سادگی در مقابل نگاشت با کیفیت بالا در مقابل نیازمندی های سیستمی کم تر). با این وجود، توانایی انطباق کامل در زمان اجرا یک نکته کلیدی در سیستم های پردازنده شبکه در آینده است.

7. نتیجه گیری

ما یک بحث کیفی گسترده ای از مسائل مربوط به طراحی سیستم عامل های پردازنده شبکه ارائه نمودیم. برای به تصویر کشیدن ملاحظات طراحی، نتایج کمی فراهم کردیم که مبادلات عملکردی بین پارامترهای طراحی مختلف را برجسته میکنند. در نهایت، سه طرح سیستم عامل مختلف و مزایا و معایب آنها را بررسی کردیم. باور داریم که این مطالعه یک مبنای مهم برای طراحی و پیاده سازی سیستم عامل های آینده برای پردازنده های شبکه فراهم می

کند. درک مبادلات ارائه شده، طراحان سیستم عامل را به در نظر گرفتن تعاملات بین برنامه های کاربردی، ترافیک شبکه، و سخت افزار زیربنایی هدایت میکند. این امر ما را به تحقق پردازنده های شبکه به عنوان اجزای به راحتی قابل استفاده در سیستم های شبکه نزدیک می کند.

8. REFERENCES

[1] BAKER, Z., AND PRASANNA, V. K. A methodology for synthesis of efficient intrusion detection systems on FPGAs. In Proc. of the IEEE Conference on Field-Programmable Custom Computing Machines (Napa, CA, Apr. 2004).

[2] ELSON, J., AND CERPA, A. Internet content adaptation protocol (ICAP). RFC 3507, Network Working Group, Apr. 2003.

[3] EZCHIP TECHNOLOGIES LTD. NP-1 10-Gigabit 7-Layer Network Processor. Yokneam, Israel, 2002. http://www.ezchip.com/html/pr np-1.html.

[4] GAVRILOVSKA, A., SCHWAN, K., NORDSTROM, O., AND SEIFU, H. Network processors as building blocks in overlay networks. In Proc. of Hot Interconnects (Stanford, CA, Aug. 2003), ACM, pp. 83–88.

[5] GOGLIN, S. D., HOOPER, D., KUMAR, A., AND YAVATKAR, R. Advanced software framework, tools, and languages for the IXP family. Intel Technology Journal 7, 4 (Nov. 2004), 64–76.

[6] IBM CORPORATION. IBM Power Network Processors, 2000. http://www.chips.ibm.com/products/wired/communications/network processors.html.

[7] INTEL CORPORATION. Intel Second Generation Network Processor, 2002. http://www.intel.com/design/network/products/npfamily/ixp2400.htm.

[8] KOHLER, E., MORRIS, R., CHEN, B., JANNOTTI, J., AND KAASHOEK, M. F. The Click modular router. ACM Transactions on Computer Systems 18, 3 (Aug. 2000), 263–297.

[9] KOKKU, R., RICHE´, T., KUNZE, A., MUDIGONDA, J., JASON, J., AND VIN, H. A case for run-time adaptation in packet processing systems. In Proc. of the 2nd Workshop on Hot Topics in Networks (HOTNETS-II) (Cambridge, MA, Nov. 2003).

[10] LUCENT TECHNOLOGIES INC. PayloadPlusTM Fast Pattern Processor, Apr. 2000. http://www.agere.com/support/non-nda/docs/FPPProductBrief.pdf.

[11] MALLOY, B. A., LLOYD, E. L., AND SOUFFA, M. L. Scheduling DAG's for asynchronous multiprocessor execution. IEEE Transactions on Parallel and Distributed Systems 5, 5 (May 1994), 498–508.

[12] MEMIK, G., AND MANGIONE-SMITH, W. H. NEPAL: A framework for efficiently structuring applications for network processors. In Proc. of Second Network Processor Workshop (NP-2) in conjunction with Ninth International Symposium on High Performance Computer Architecture (HPCA-9) (Anaheim, CA, Feb. 2003).

[13] MOTWANI, R., AND RAGHAVAN, P. Randomized Algorithms. Cambridge University Press, New York, NY, 1995.

[14] PLISHKER, W., RAVINDRAN, K., SHAH, N., AND KEUTZER, K. Automated task allocation for network processors. In Proc. of Network System Design Conference (Oct. 2004), pp. 235–245.

[15] RAMASWAMY, R., WENG, N., AND WOLF, T. Application analysis and resource mapping for heterogeneous network processor architectures. In Proc. of Third Workshop on Network Processors and Applications (NP-3) in conjunction with Tenth International Symposium on High Performance Computer Architecture (HPCA-10) (Madrid, Spain, Feb. 2004), pp. 103–119.

[16] RAMASWAMY, R., WENG, N., AND WOLF, T. Analysis of network processing workloads. In Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (Austin, TX, Mar. 2005), pp. 226–235.

[17] SHAH, N., PLISHKER, W., AND KEUTZER, K. NP-Click: A programming model for the intel IXP1200. In Proc. of Second Network Processor Workshop (NP-2) in conjunction with Ninth International Symposium on High Performance Computer Architecture (HPCA-9) (Anaheim, CA, Feb. 2003), pp. 100–111.
[18] TAYLOR, D. E., TURNER, J. S., LOCKWOOD, J. W., AND HORTA, E. L. Dynamic hardware plugins: Exploiting reconfigurable hardware for high-performance programmable routers. Computer Networks 38, 3 (Feb. 2002), 295–310.
[19] TEJA TECHNOLOGIES. TejaNP Datasheet, 2003. http://www.teja.com.
[20] WENG, N., AND WOLF, T. Profiling and mapping of parallel workloads on network processors. In Proc. of The 20th Annual ACM Symposium on Applied Computing (SAC) (Santa Fe, NM, Mar. 2005), pp. 890–896.
[21] WOLF, T., AND FRANKLIN, M. A. CommBench - a telecommunications benchmark for network processors. In Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) (Austin, TX, Apr. 2000), pp. 154–162.