به نام خدا

تمرین اول

علیرضا آخوندی

9731107

سوال اول:

تاخیر بالا در برای دسترسی به خانه های حافظه: با این که زمان دسترسی به داده مورد نظر کم به نظر می‌رسد اما این زمان با توجه به سرعت بسیار بالای پردازنده کم است.این عامل یک گلوگاه برای اجرای دستورالعمل های fetch شده است. برای مثال فرض کنید که زمان دسترسی به یک خانه حافظه ۱۰۰ cycle بوده و درحالی که پردازنده ما یک پردازنده issue 4 بوده که توانایی اجرای ۴ دستور را در یک cycle دارد. از عوامل این تاخیر می تواند فاصله بین پردازنده با حافظه یا پهنای باند باشد.کوچک بودن پهنای باند ممکن است نتواند نیاز پردازنده را برطرف کند از طرفی بزرگی زیاد آن نیز ممکن است access time را افزایش دهد که این خود نیز یک گلوگاه است.

سوال دوم:

حافظه نهان : استفاده از Cache یا حافظه نهان می‌تواند تا حد خوبی این مشکل را حل کند. ایده اصلی این است که با ذخیره سازی داده هایی که پردازنده بیشتر برای استفاده از آن ها درخواست می‌فرستد در یک حافظه سریع تر در نزدیکی پردازنده ، از درخواست های مکرر به RAM جلوگیری کنیم.اضافه کردن سیاست Locality of reference ، که در آن اگر درخواستی برای دسترسی به یک خانه حافظه داده شود خانه های اطراف آن خانه از حافظه را نیز به امید اینکه مورد استفاده قرار بگیرند به حافظه نهان می آوریم ، نیز می‌تواند مفید باشد.

پیش‌واکشی : در این روش هنگامی که در حال اجرای یک دستور هستیم ، داده های مربوط به دستور بعدی که مستقل از دستور در حال اجرا هست را به حافظه نهان می‌آوریم

اجرای چند نخی: در این روش هنگامی که یک برنامه یا نخ منتظر رسیدن داده مورد نیاز خود است ، منطقی است که اجرای یک فرایند یا نخ دیگر را شروع کنیم یا ادامه دهیم.

Locality of Reference روشی است که در بیشتر سیستم های امروزی(چه سریال چه موازی) استفاده می شود.

سوال سوم:

۱. چند نخی درشت دانه: در این روش بین هر نخ تنها زمانی در انتظار (stall) انجام یک عملیات وقتگیر (مانند آوردن داده از حافظه) هستند ، context switch رخ می دهد.

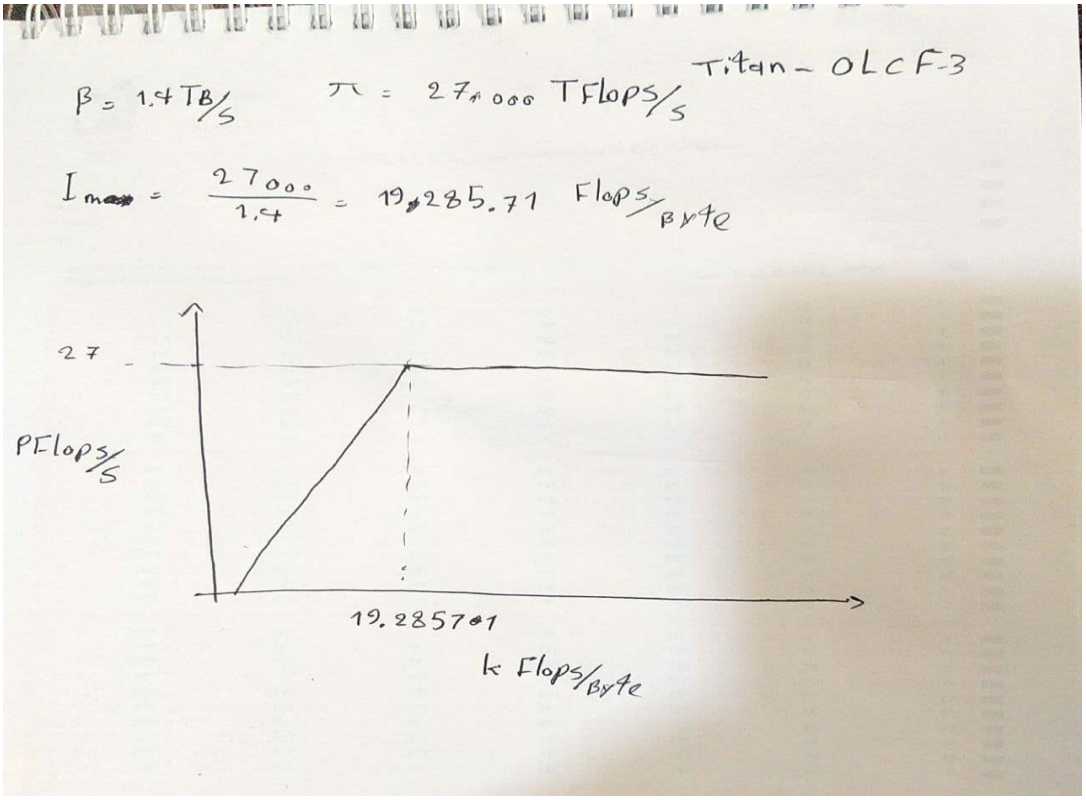
۲. چند نخی ریز دانه: در این روش در هر cycle پردازنده به یک نخ اختصاص میابد.این روش با این که هدر رفت منابع را کاهش میدهد ، اما در این روش فرایند هایی که یک دنباله طولانی از دستورات را باید انجام دهند ، باید بعد از انجام هر دستور ، منابع را به فرایند دیگری بدهند.

۳. چند نخی همزمان: ایده اصلی این روش استفاده از چند نخ به طور همزمان و تنظیم دستورات آن برای اجرا در یک cycle است.به این صورت که در یک cycle ، هر issue به یک نخ اختصاص یابد

امکان هدر رفت منابع در چند نخی همزمان بسیار کمتر است.

سوال چهارم:

در این مدل طبق محاسباتی بر حسب بیشینه کارایی پردازشی هسته ها، بیشینه پهنای باند و ضرب شدت حسابی برنامه میتوانیم محاسبه کنیم که کدام بخش از سیستم برای یک برنامه به گلوگاه تبدیل میشود. به این صورت که اگر در هر ثانیه β بایت از حافظه بتوانیم بخوانیم و به ازای هر بایت خوانده شده بخواهیم I عملیات ممیز شناور انجام دهیم، در کل $\beta \times I$ عملیات ممیز شناور میتوانیم انجام دهیم. در این صورت اگر $\beta \times I$ از بیشینه کارایی پردازشی هسته‌ها یا همان π بیشتر باشد، پردازنده به گلوگاه تبدیل میشود و در صورتی که از این مقدار کوچکتر باشد پهنای باند به گلوگاه تبدیل خواهد شد.



سوال پنجم:

طبقه بندی فلین ، یک طبقه بندی معروف برای معماری کامپیوتر ها بر اساس تعداد جریان های همزمان (Concurrent) داده و دستورات موجود در معماری است.

Single instruction stream, single data stream (SISD):

در این نوع سیستم ها هیچ گونه موازی سازی دیده نمی‌شود. تنها یک واحد پردازنده وجود دارد که در هر قدم دستور را از حافظه خوانده و داده مربوطه به آن را نیز از حافظه می‌خواند. نتیجه نیز در حافظه ذخیره می‌شود.مثال هایی از این سیستم:

IBM 701, IBM 1620 IBM 7090

Single instruction stream, multiple data streams (SIMD) :

در این سیستم ها چندین جریان داده را داریم که می خواهیم یک دنباله ای از دستورات را بر رو آن ها به صورت موازی انجام دهیم. در این سیستم ها در هر قدم یک دستور بر روی چند جریان داده اعمال می شود. در کاربرد هایی با درجه بالای موازی سازی داده SIMD روش مفیدی است. برای مثال کامپیوتر هایی با پردازش های سنگین گرافیکی برای تولید فضای سه بعدی واقعیت مجازی ، می توان از این روش استفاده کرد.

Illiac-IV

PEPE
BSP

:Multiple instruction streams, single data stream (MISD)

در این روش چندین واحد پردازنده وجود دارد که به یک حافظه مشترک global دسترسی دارند. در این روش هر واحد پردازش روی یک داده مشخص ، مجموعه ای از دستورات را انجام میدهند. در واقع داده ای که به هر واحد پردازش داده می شود یکسان است اما دستورات هر واحد متفاوت است.این مدل بیشتر برای آزمایش و به تعداد محدود ساخته شده است و تا به حال هیچ کامپیوتر با این مدل به خط تولید نرفته است.

:Multiple Instruction, Multiple Data (MIMD)

در این مدل تعدادی واحد پردازشی وجود دارند که هرکدام دسترسی دستور و حافظه جدا به یک برنامه یا حافظه (مشترک یا توزیع شده) را دارد. در هر قدم هر واحد دستور و داده خود را جداگانه و به صورت asynchronous نسبت به واحد های دیگر load می کند و نتیجه را در حافظه ذخیره می کند.
سیستم های چند پردازنده و یا خوشه ها نمونه ای MIMD ها هستند.
IBM 370/168 MP, Univac 1100/80

مقایسه MIMD و SIMD :

مدل SIMD این مزیت را نسبت به MIMD دارد که راحت تر پیاده سازی و program می‌شود و تنها یک program flow داریم ، همگام سازی در سطح برنامه نیاز نداریم. اما همگام سازی در اجرای برنامه نیاز است. همگام سازی در زمان اجرا در زمان بر خوردن به دستورات شرطی لازم است.

برای مثال در کد زیر :

```
if (b==0)
        c = a;
else
        c = a/b;
```

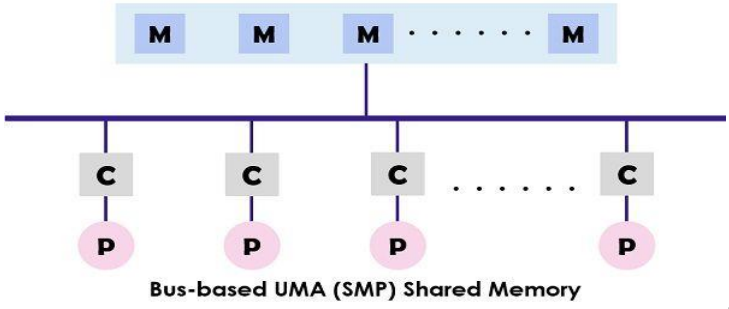
برای اجرای این کد به دو قدم نیاز است. در قدم اول ، تمام واحد های پردازشی که متغیر محلی b در آن ها صفر است بخش IF را اجرا می کنند و در قدم دوم تمام فرایند هایی که b در آن ها صفر نیستند اجرا می شوند.

سوال 6:

Shared Memory: در این روش برای ارتباط بین دو یا چند واحد پردازشی ، یک حافظه مشترک ایجاد می شود.این روش روش ارتباطی سریعی است اما نیاز به مکانیزم های همگام سازی برای کنترل خواندن و نوشتن بر روی این حافظه مشترک داریم تا از race condition جلوگیری کنیم.

Message Passing: در این روش ارتباط بین واحد های پردازشی از طریق ارسال پیام صورت می گیرد. در این روش از یک واسط برای ارسال پیام بین واحد های پردازشی استفاده می شود.این روش ارتباط از روش قبلی کندتر است اما از آنجایی ایجاد حافظه مشترک بین کامپیوتر هایی با سخت افزار مجزا سخت است (یا غیر ممکن) از این روش استفاده می کنیم. در این روش هر واحد پردازنده ها و حافظه های شخصی خودشان دارند.

1) در این معماری هر واحد پردازش حافظه محلی خود با فضای آدرس دهی شخصی داشته و ارتباط بین این واحد ها از طریق ارسال پیام صورت می‌گیرد. این سیستم ها از تعدادی حافظه و پردازنده تشکیل شده اند که طریق یک ارتباط داخلی به هم مربوط می شوند. این ارتباط می تواند توسط point to point link ها یا یک سخت افزار جدا برای جابه جایی network انجام شود.
2) 1- Uniform Memory Access: در این معماری تمامی پردازنده ها از یک حافظه مشترک استفاده می کنند. همان طور که از اسم این معماری پیداست زمان دسترسی تمامی واحد های

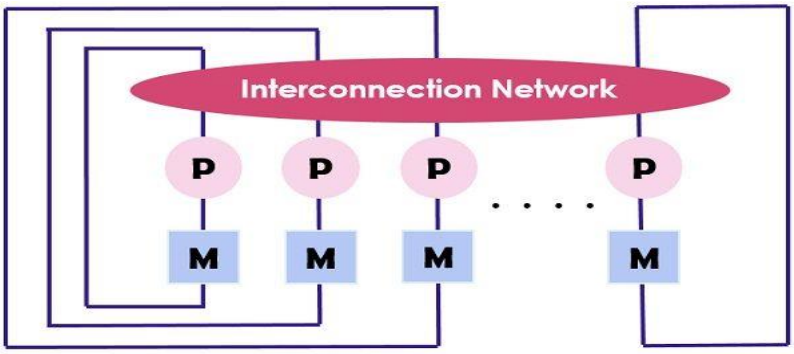


پردازش برابر است.

همان طور که در شکل دیده می شود هر واحد پردازش ابتدا به حافظه نهان خود ، سپس به bus و در انتها به حافظه مشترک متصل می شود. مثال هایی از این سیستم ها :

Sun Starfire servers, Compaq alpha server and HP v series.

2- Non-uniform Memory Access : در این معماری هر واحد پردازش حافظه مخصوص خود را دارد با این حال فضای آدرس دهی تمام این حافظه ها یکسان است.تفاوت این معماری با معماری قبلی این است که زمان دسترسی به خانه های حافظه به فاصله حافظه با پردازنده دارد و برای همه یکسان نیست.



NUMA Shared Memory System

BBN, TC-2000, SGI Origin 3000, Cray

مثال های از این معماری هستند.

سوال هفت:

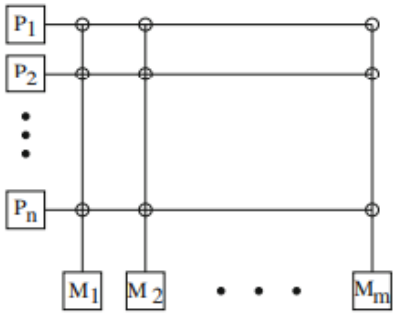
Shared Bus:

یک bus از مجموعه ای از سیم ها تشکیل شده است که برای ارسال داده از فرستنده به گیرنده استفاده می‌شوند. بعضی مواقع هزاران سیم برای اطمینان از سرعت بالا ارسال داده استفاده می‌شود. در هر مقطع زمانی فقط یک انتقال داده می‌تواند صورت بگیرد. هنگامی که چند واحد پردازشی بخواهند به طور همزمان داده را انتقال دهند ، واحدی به نام bus arbiter برای سازمان دهی این انتقال ها استفاده می‌شود.این مکانیزم ارتباط ساده و ارزان است اما مقیاس پذیری کمی دارد و برای مواقعی مفید است که تعداد واحد های پردازشی کم است.

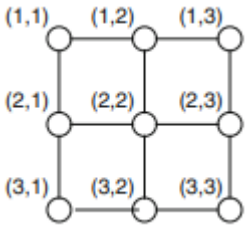
مثال: Intel Pentium Pro 4-processor quad-pack

Crossbar: شکل زیر نشان دهنده یک شبکه Crossbar است. تمام واحد های پردازشی امکان اتصال مستقیم به تمامی حافظه ها (یا واحد های پردازشی دیگر در سیستم های توزیع شده) را دارند. این ارتباط از طریق سوییچ های قرار گرفته در شبکه امکان پذیر می‌شود. این روش ارتباط برای شبکه ها با تعداد واحد های کم مفید است اما برای کاربرد هایی با تعداد زیاد واحد پردازشی به علت سخت افزار زیاد بهینه نیست.

مثال: UltraSPARC T2



Mesh: یک شبکه ای از واحد های عملیاتی که به صورت مستقیم ، پویا و غیر سلسله مراتبی به یک دیگر متصل هستند و با یک دیگر برای انتقال داده همکاری می کنند. اتصال این واحد های پردازشی لزوما مستقیم نیست و ممکن است در مسیر ارتباطی دو واحد ، چندین واحد دیگر قرار داشته باشند.این روش نسبت به روش های قبلی dynamic تر و مقیاس پذیر تر است و می توان از آن برای شبکه های بزرگ استفاده کرد. شکل زیر نشان دهنده یک مش دو بعدی است:

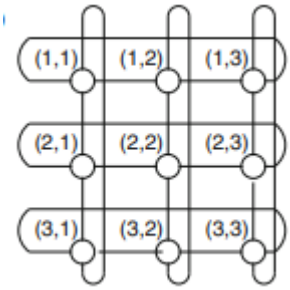


مثال هایی از این شبکه ارتباطی:

Intel SCC

Intel Xeon Scalable Processor

Torus: نوع تغییر یافته ای از شبکه های مش است. یک d-dimensional torus یک نوع مش است که گره های اول و آخر هر بعد آن به یک دیگر متصل هستند. شکل زیر نمونه ای از 2-deimensional torus است :



مثال :

IBM BlueGene

Cray XT3, XT4, and XT5

در 2d-torus به دلیل متصل بودن گره های انتهایی هر بعد ، ارتباط بین ها سریع تر است اما در 2d-mesh ارتباط بین گره های انتهایی هر بعد هزینه بیشتری می برد زیرا باید کل گره های بین این دو گره پیمایش شوند که این در شبکه های با واحد های زیاد بسیار کند و هزینه بر می شود. اما این پیاده سازی torus سخت افزار و هزینه بیشتری لازم دارد. یک شبکه مش با N گره به N-1 اتصال نیاز دارد در حالی که torus با همان تعداد گره به 2N ارتباط نیاز دارد.

سوال هشت :

تسریع : معیاری برای اندازه گیری تاثیر موازی سازی بر اجرای یک برنامه است. سرعت اجرای برنامه سریال (در بهترین پیاده سازی خود) به سرعت اجرای موازی برنامه را تسریع می‌گوییم.

بهره وری : این معیار به ما می‌گوید که آیا تسریع حاصل شده ، با هزینه منطقی ای انجام شده یا خیر.
تسریع بر تعداد هسته ها نشان دهنده این مقدار است

مقیاس پذیری : این معیار نشان دهنده عملکرد و رفتار یک برنامه موازی با بزرگ شدن مسئله است. فرض کنید یک برنامه موازی با اندازه ثابت و با تعداد معلوم نخ اجرا شده و بازده E به‌دست آمده است. حال تعداد

نخ‌ها را افزایش می‌دهیم. اگر بتوان مقداری برای نرخ افزایش اندازه مسأله یافت که بازده ثابت بماند، سیستم مقیاس‌پذیر است.

1) تسریع فوق خطی زمانی اتفاق می افتد که s > p باشد. این تسریع معمولا به طور شانسی و به ندرت اتفاق می افتد. بعضی اوقات حافظه نهان می تواند تاثیرگذار باشد. برای سیستم هایی که هر پردازنده حافظه نهان مخصوص به خود را دارد این اتفاق ممکن است رخ دهد. یک برنامه سریالی که بر روی یک پردازنده اجرا می شود را در نظر بگیرید، این برنامه با حجم زیادی از داده سر و کار دارد و از آنجایی که حافظه نهان ما محدود است تعداد cache miss ها نیز بالا می رود.اما همین برنامه را اگر موازی سازی کنیم ممکن است داده های مورد نیاز برنامه به طور مناسبی بین حافظه های نهان هر پردازنده پخش شوند و یک توزیع مناسب باعث شود تسریع ما فوق خطی شود.
2) انواع تسریع عبارت‌اند از : تسریع فوق خطی – تسریع خطی – تسریع زیر خطی. عوامل موثر بر تسریع عبارت اند از : هنر برنامه‌نویس - انتخاب کامپایلر - تنظیمات کامپایلر - سیستم عامل - نوع سیستم فایل - میزان بار کاری سیستم
3) خیر. این اتفاق همیشگی نیست .طبق قانون آمدال اضافه کردن تعداد هسته تا یک تعداد مشخصی مفید است. از یکجایی به بعد تسریع یا ثابت می ماند یا کاهش میابد. در واقع تعداد هسته ها تنها عامل محدود کننده تسریع نمی تواند باشد. بخشی از آن بسته به درجه موازی سازی الگوریتم است.حتی اگر درجه موازی سازی 1 هم باشد (تسریع خطی) نمی توان انتظار داشت که تسریع n برابر شود.تسریع قبلی p بوده و تسریع جدید n است در نتیجه $\frac{n+p}{p}$ نسبت تسریع قبلی به تسریع جدید خواهد بود. تنها در ایده آل ترین حالت اگر تسریع فوق خطی داشته باشیم ممکن است n برابر شود که احتمال آن بسیار کم است.
4) با افزایش اندازه مساله تسریع و بهره وری افزایش میابد زیرا اندازه بار کاری هر نخ زیاد شده و overhead مربوط به موازی سازی درصد کمتری از زمان اجرا را به خود اختصاص می دهد.
5) هرچه تسریع بیشتر باشد، بهره‌وری نیز بیشتر خواهد بود.
6) مقیاس پذیری رابطه‌ی مستقیمی با بهره وری و تسریع ندارد.
7) مقیاس پذیری به خصوصیاتی از الگوریتم و اجرای موازی آن بستگی دارد. در واقع مقیاس پذیری به برسی این خصوصیت از پیاده سازی الگوریتم موازی ما می پردازد که آیا می توان بهره وری را با افزایش تعداد هسته ها و اندازه مساله ثابت نگه داشت یا خیر.
8)

سوال نه :

وقتی در برنامه ای بهبودی از طریق اضافه کردن واحد های موازی حاصل می شود ، لزوما این بهبود روی کل برنامه حاصل نمی شود.این بهبود روی کسری از برنامه که قابلیت موازی سازی دارد حاصل می شود. تسریع حاصل از این بهبود را قانون آمدال می گویند.

قانون گوستاوسون یک تسریع در حد تئوری از اجرای یک تسک با زمان اجرای موازی ثابت که تنها از ماشین های بهبود یافته توقع انجام آن می رود.

تفاوت این دو قانون در این است که در قانون آمدال اندازه مساله را ثابت در نظر می گیریم و سعی در بهبود ماشین خود داریم این در حالی است که قانون گوستاوسون فرض بر این است که ماشین انجام دهنده تسک بهبود یافته است و در اینجا اندازه مساله و تعداد پردازنده ها متغیر است (به طوری که زمان اجرای موازی الگوریتم ثابت بماند)

اگر مساله در حالت سریال Ts واحد زمانی طول بکشد و a درصد آن قابل موازی سازی و 1-a درصد آن غیر قابل موازی سازی باشد و برای موازی سازی از p هسته استفاده کنیم، حداکثر تسریع مساله را در حالت موازی از رابطه زیر به دست می‌آوریم:

$$S = \frac{Ts}{Tp} = \frac{Ts}{\frac{aTs}{p} + (1-a)Ts} = \frac{1}{\frac{a}{p} + 1 - a}$$

سوال ده :

هرگاه بتوانیم با افزایش تعداد واحد های پردازش و ثابت نگه داشتن اندازه مساله بهره وری را ثابت نگه داریم آنگاه می گوییم برنامه ما قویا مقیاس پذیر است. اگر یک برنامه تسریع خطی داشته باشد آنگاه طبق رابطه زیر بهره وری آن ثابت و برابر یک خواهد بود که همان شرط ما است.

$$E = \frac{Ts}{pTp} = \frac{Ts}{p \times \frac{Ts}{p}} = 1$$