دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

برنامه‌نویسی چندهسته‌ای

تمرین اول: آشنایی با معماری سیستم‌های موازی

رادین شایانفر

شماره دانشجویی: ۹۷۳۱۰۳۲

بهار ۱۴۰۰

۱. محدودکننده‌های کارآیی حافظه در سیستم‌های موازی را نام برده و شرح دهید هر کدام چگونه باعث محدودیت در کارآیی حافظه می‌شوند.

در سیستم‌های موازی معمولا پردازنده‌ها محدودکننده کارایی نیستند و تاخیر (Latency) زیاد و پهنای باند (Bandwidth) پایین حافظه‌ها، آن‌ها را تبدیل به گلوگاه می‌کند.

تاخیر بالا باعث می‌شود از زمانی که پردازنده درخواست دریافت داده‌ای را می‌دهد تا زمانی که به حافظه می‌رسد و پردازش می‌شود مدت زمان زیادی طول بکشد (به عنوان مثال فاصله‌ی زیاد حافظه و پردازنده از عوامل بالا رفتن تاخیر است). در این زمان ممکن است پردازنده بی‌کار بماند که باعث پایین آمدن کارایی آن (CPU Utilization) می‌شود.

از طرف دیگر پهنای باند پایین بر روی مقدار اطلاعاتی که پردازنده در واحد زمان از حافظه می‌تواند دریافت کند تاثیر می‌گذارد. در واقع اگر داده‌هایی که پردازنده در واحد زمان از حافظه نیاز دارد از پهنای باند حافظه بالاتر باشد، موجب بی‌کار ماندن آن نخ می‌شود که می‌تواند روی کارایی پردازنده تاثیر بگذارد.

---

۲. یکی از چالش‌های موجود در سیستم‌های موازی تاخیر حافظه می‌باشد. راهکارهای پنهان‌سازی این تاخیر را نام برده و شرح دهید. همچنین بیان کنید امروزه در سیستم‌های موازی از کدام راهکارها بیشتر استفاده می‌شود.

برای پنهان‌سازی تاخیر حافظه، از روش‌های زیر می‌توان استفاده کرد:

۱- محلی بودن ارجاع (Locality): همان‌طور که می‌دانیم طبق اصل همجواری مکانی، معمولا درخواست‌ها به حافظه در یک برنامه دارای همجواری مکانی هستند. در نتیجه با روش‌هایی (مانند آوردن و کش کردن داده‌ها به صورت بلوکی در حافظه نهان) تاخیر حافظه را کم‌اثر کرد.

۲- پیش‌واکشی (Prefetching): در این روش، اگر مکانیزمی (سخت‌افزاری یا نرم‌افزاری) برای واکشی داده‌ها مدتی قبل از آنکه واقعا به آن‌ها نیاز باشد وجود داشته باشد، می‌توان با این کار اثر تاخیر حافظه را کاهش داد.

۳- اجرای چندنخی (Mutlithreading): این راهکار با اجرای چندین نخ، باعث می‌شود زمانی که یکی از نخ‌ها به داده‌ای از حافظه نیاز دارد، پردازنده بتواند به سراغ اجرای نخ‌های دیگر برود و زمان را هدر ندهد.

هر سه این روش‌ها کمک زیادی به افزایش سرعت سیستم‌های موازی می‌کند. روش اول تقریبا در همه‌ی این سیستم‌ها استفاده می‌شود. روش دوم ممکن است در برخی پردازنده‌ها به صورت سخت‌افزاری پیاده‌سازی شده باشد. به صورت نرم‌افزاری نیز بسته به کامپایلر امکان استفاده از آن وجود دارد. روش سوم اما به نظر از دو روش دیگر کارآمدتر است و از آنجا که استفاده از آن در دست برنامه‌نویس است، استفاده بسیاری از آن در سیستم‌های امروزی می‌شود.

---

۳. مکانیزم‌های مختلف چندنخی را نام برده و هر کدام را شرح دهید. همچنین تحقیق کنید کدام مکانیزم‌ها در سیستم های موازی امروزی کاربرد دارند و کدام مکانیزم نسبت به بقیه مکانیزم‌ها برتری دارد. دلایل خود را با ذکر مثال شرح دهید.

مکانیزم‌های مختلف چندنخی عبارتند از:

۱- درشت‌دانه (Coarse Grain): در این حالت هر نخ چندین سیکل اجرا می‌شود و سپس با context switch نوبت به نخ بعدی می‌رسد.

۲- ریزدانه (Fine Grain): در چندنخی ریزدانه زمان context switch به حداقل رسیده است. به طوری که هر نخ می‌تواند تنها در یک سیکل اجرا شود و بلافاصله در سیکل بعدی نخ دیگری اجرا شود.

۳- همزمان (SMT یا Hyperthreading در پردازنده‌های اینتل): در این روش در هر سیکل چندین نخ می‌توانند همزمان با هم اجرا شوند. به این معنی که پردازنده قادر به ذخیره وضعیت چندین نخ به طور همزمان است و نیازی به context switch بین آن‌ها نیست. در نتیجه نخ‌های مختلف می‌توانند همزمان در یک سیکل از واحدهای پردازشی مختلف یک پردازنده (مثلا یک پردازنده 4-issue) استفاده کنند.

در بین ۳ راهکار بالا، راهکار دوم، که در barrel processorها به کار می‌رود، تقریبا امروزه استفاده نمی‌شود. هم‌چنین Hyperthreading که از اوایل سال ۲۰۰۲ به پردازنده‌های اینتل (Pentium 4 و Xeon) اضافه شدند، استفاده از آن در چند سال گذشته کم‌کم در حال کاهش است. در نتیجه چندنخی درشت‌دانه در حال حاضر بیشتر کاربرد دارد.

در چندنخی ریزدانه زمان‌هایی که نخ‌ها بیکار هستند تا حد خوبی نسبت به حالت درشت‌دانه پنهان می‌شوند. هر چند زمان اجرای هر نخ طولانی‌تر می‌شود. در چندنخی درشت‌دانه نیازی به سرعت بسیار بالا برای context switch نیست، هر چند هر بار context switch باعث خالی شدن کامل پایپ‌لاین می‌شود.

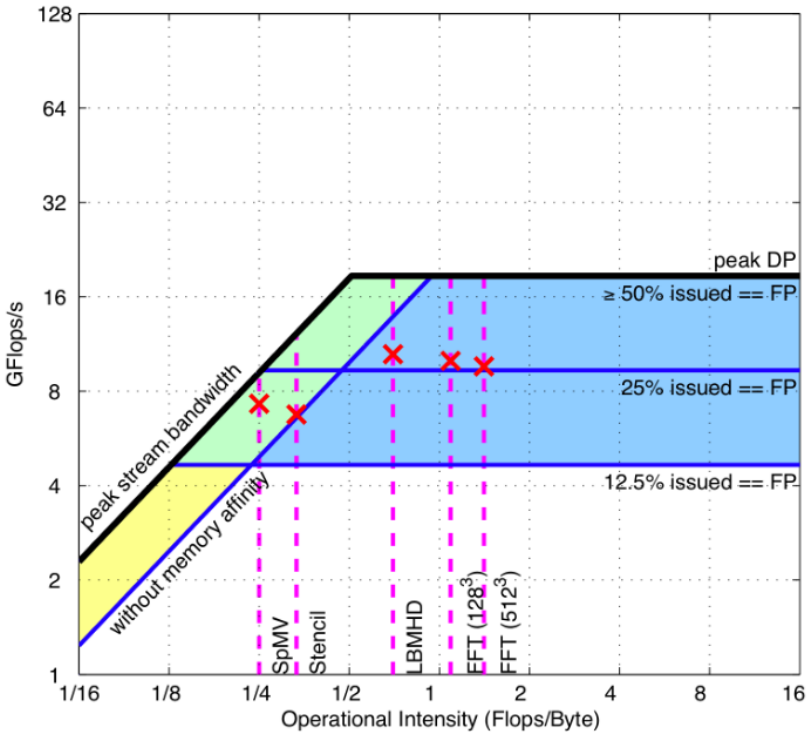
حالت SMT به نظر از دو روش دیگر بهتر است زیرا از آنجا که یک نخ به سختی می تواند از تمام توان پردازنده استفاده کند، این روش بیشترین استفاده را از توان پردازشی پردازنده می‌برد.

---

۴. مدل کارآیی پشت بام را شرح دهید. همچنین یک سیستم چندهسته‌ای در دنیای واقعی را به دلخواه در نظر بگیرید و کارآیی آن را با مدل پشت بام نمایش دهید.

در این مدل دیگر محدودیت‌های سیستم مانند پهنای باند حافظه نیز در کنار کارایی هسته‌ها نمایش داده می‌شود. به طوری که توسط آن می‌توان تخمینی از گلوگاه سیستم داشت که مثلا مرتبط با حافظه است یا پردازنده.

به عنوان مثال در مقاله این مدل، نمودار پردازنده Sun UltraSPARC T2+ رسم شده است و در آن اجرای چندین الگوریتم مقایسه شده است. همان‌طور که می‌بینیم الگوریتمی مانند Sparse Matrix-Vector Multiplication در این سیستم memory-bound و تبدیل فوریه سریع (FFT) برنامه‌ای compute-bound در این سیستم است.



۵. طبقه بندی فلین را توضیح دهید. سپس به سوالات زیر پاسخ مناسب دهید.

۱) طبقه بندی فلین شامل چه دسته هایی می باشد؟ این دسته ها را با هم مقایسه کنید و برای هر کدام چند مثال از دنیای واقعی ذکر کنید.

۲) تفاوت معماری MIMD و SIMD چیست؟ آیا این معماری‌ها دارای محدودیت‌هایی می باشند؟ در صورت مثبت بودن، پاسخ خود را شرح دهید.

طبقه‌بندی فلین برای طبقه‌بندی معماری کامپیوترها در سال ۱۹۶۶ توسط مایکل فلین معرفی شد.

۱) این طبقه‌بندی شامل ۴ دسته‌بندی SISD، SIMD، MIMD و MISD می‌باشد. این دسته‌بندی‌ها مربوط به معماری جریان داده‌ها و دستورالعمل‌ها می‌باشد.

- SISD: در این معماری تنها یک دستورالعمل در لحظه بر روی یک جریان داده اجرا می‌شود. به عنوان مثال پردازنده‌های تک‌هسته‌ای قدیمی با معماری فون‌نویمان از این دسته هستند.
- SIMD: در معماری SIMD یک دستورالعمل به طور همزمان بر روی چند جریان داده مختلف اجرا می‌شود. به عنوان مثال هسته‌های کارت‌های گرافیک (با نام SIMT در پردازنده‌های NVIDIA) از این معماری استفاده می‌کنند.
- MIMD: در این معماری چندپردازنده به طور مستقل دستورالعمل‌ها را بر روی جریان‌های داده‌ی متفاوت اجرا می‌کنند. سیستم‌های چندهسته‌ای سوپراسکالر، خوشه‌ها و سیستم‌های توزیع شده از این دسته هستند.
- MISD: این دسته که کمتر از دسته‌های دیگر متداول است در سیستم کنترل پرواز شاتل‌های فضایی استفاده می‌شد.

۲) در معماری SIMD یک واحد کنترل (Control Unit) و یک دیکودر در آن، چندین المان پردازشی (Processing Element) را کنترل می‌کند. در نتیجه تنها یک دستورالعمل واحد می‌تواند واکشی و دیکود شود و به طور همزمان بر روی آن‌ها اجرا شود. به عنوان مثال واحدهای SSE و AVX در پردازنده‌های اینتل که مربوط به عملیات برداری هستند از این دسته هستند.

در معماری MIMD هر المان پردازشی، یک واحد کنترل و یک دیکودر مختص به خود دارد. در نتیجه هر کدام به صورت جداگانه می‌توانند دستورالعمل مربوط به خود را واکشی و دیکود کرده و سپس اجرا کنند.

معماری SIMD دارای محدودیت در مواجهه با دستورات شرطی است. به این صورت که در این معماری همه‌ی واحدهای پردازشی باید یک دستورالعمل را اجرا کنند، اما در دستورات if/else ممکن است جریان اجرای برنامه برای واحدهای مختلف با توجه به داده‌ای که پردازش می‌کنند متفاوت شود. در نتیجه باید شرط‌ها در سیکل‌های متفاوت اجرا شوند که کمی باعث کاهش کارایی می‌شود. این محدودیت اما در معماری MIMD وجود ندارد. هر چند پیچیدگی مداری MIMDها باعث مقیاس‌پذیری پایین آن‌ها می‌شود. (منبع)

---

۶. انواع معماری‌های موازی بر اساس مدل ارتباطی را نام ببرید و هر کدام را به طور کامل شرح دهید. سپس به سوالات زیر پاسخ دهید.

۱) معماری مشترک توزیعی با مکانیزم تبادل پیام را با ذکر مثال شرح دهید.

۲) از دید پردازنده و از نظر زمان دسترسی به حافظه چند نوع حافظه مشترک داریم؟ نام برده و هر کدام را کامل شرح دهید. بررسی کنید امروزه چه پردازنده های از چه نوع از حافظه های مشترک استفاده می کنند.

مدل‌های ارتباطی دارای ۲ نوع حافظه مشترک (Shared Memory) و تبادل پیام (Message Passing) هستند.

در مدل حافظه مشترک، یک پردازنده داده‌ای را در یک حافظه‌ای که با پردازنده‌ی دیگر مشترک است می‌نویسد و پردازنده(های) دیگر از آن می‌خواند. این معماری معمولا در سیستم‌هایی که فضای آدرس‌دهی پردازنده‌ها مشترک است قابل استفاده است. به عنوان مثال پردازنده‌های چندهسته‌ای که در یک ماشین هستند می‌توانند از این روش استفاده کنند. این روش به دلیل نداشتن سربار فراخوانی‌های سیستمی سرعتی بالاتری نسبت به روش تبادل پیام دارد. هر چند که استفاده از آن باید کاملا توسط برنامه‌نویس مدیریت شود.

مدل تبادل پیام به جای نوشتن پیام در حافظه، آن را توسط شبکه ارتباطی بین دو پردازنده ارسال می‌کند. از این روش بیشتر در سیستم‌های توزیع شده که ماشین‌ها و پردازنده‌ها دور از هم و از طریق یک شبکه ارتباطی به یکدیگر متصل هستند استفاده می‌شود. هر چند که این روش کدنویسی کمتری توسط برنامه‌نویس لازم دارد، اما به دلیل سربارهای ارسال و دریافت پیام از روش قبل کندتر است.

۱) در این معماری حافظه‌هایی که به طور فیزیکی جدا از هم هستند، همگی از طریق یک فضای آدرس‌دهی مشترک توسط چند پردازنده قابل دسترسی هستند. به عنوان مثال سیستم حافظه مشترک توزیعی مبتنی بر صفحه (Page Based Distributed Shared Memory) به نام IVY که در سال ۱۹۸۶ ارائه شد از این مدل است. در این سیستم تعدادی کلاینت از طریق شبکه محلی (LAN) به یک سرور متصل هستند. هر یک از این کلاینت‌ها دارای حافظه جداگانه هستند که توسط کلاینت‌های دیگر از طریق شبکه محلی قابل دسترسی است.

۲) از این نظر دو نوع حافظه مشترک داریم:

- حافظه مشترک با دسترسی حافظه یکنواخت (UMA): در این نوع حافظه دسترسی همه‌ی پردازنده‌ها به همه جای حافظه به یک اندازه زمان می‌برد.
- حافظه مشترک با دسترسی حافظه غیریکنواخت (NUMA): در مدل NUMA قسمتی از حافظه که به یک پردازنده نزدیک‌تر است، سرعت دسترسی بیشتری نسبت به قسمت‌های دیگر حافظه برای آن پردازنده دارد.

با بزرگ‌تر شدن حافظه‌ها، ایجاد دسترسی یکنواخت (UMA) به طوری که زمان دسترسی همچنان کوتاه باشد مشکل می‌شود. در نتیجه با غیریکنواخت کردن دسترسی (NUMA)، سرعت دسترسی به آن بخشی از حافظه که به یک پردازنده نزدیک‌تر است از حالت UMA بیشتر خواهد بود و این مزیت NUMA نسبت به UMA است. در نتیجه از UMA بیشتر در سیستم‌های عام منظوره و time-sharing و از NUMA در سیستم‌های بی‌درنگ و با حساسیت بالا یا سیستم‌هایی که اندازه حافظه امکان استفاده از UMA را به دلیل کندی بالا نمی‌دهد استفاده می‌شود.

---

۷. انواع شبکه های میان ارتباطی را نام برده، مزایا و معایب آن ها را با ذکر دلیل بیان کنید. سپس به سوالات زیر پاسخ دهید.

۱) هر کدام از شبکه های میان ارتباطی امروزه در چه پردازنده‌هایی استفاده می‌شوند؟

۲) در مورد هزینه ارتباطی مش دوبعدی و توری دو بعدی تحقیق کنید.

انواع شبکه‌های میان ارتباطی (به همراه استفاده آن‌ها) عبارتند از:

- گذرگاه مشترک (Shared Bus): این شبکه ساده‌ترین و ارزان‌ترین نوع است. اما کارایی و مقیاس‌پذیری پذیری پایینی از نظر پهنای باند دسترسی به حافظه دارد. زیرا با افزایش تعداد پردازنده‌ها، پهنای باند حافظه تبدیل به گلوگاه سیستم می‌شود. پردازنده Intel Pentium Pro 4-processor quad-pack از این شبکه ارتباطی استفاده می‌کند.
- کراس بار (Crossbar): کراس بار به دلیل دادن اجازه دسترسی همزمان چند پردازنده به قسمت‌های مختلف حافظه سرعت بالایی دارد. اما هزینه پیاده‌سازی این نوع شبکه بالاست و با افزایش تعداد پردازنده‌ها و پیچیده شدن آن، امکان مقیاس‌پذیری به دلیل هزینه وجود ندارد. شبکه کراس بار در شبکه‌های core-to-cache-bank مانند IBM POWER5 و Sun Niagara I/II استفاده می‌شود.
- حلقه (Ring): این شبکه در سیستم‌های پیچیده‌تر کارایی بهتری از باس مشترک دارد. همچنین نیازی به یک گره مرکزی برای کنترل اتصال ندارد و مسیریابی در آن به سادگی صورت می‌گیرد. اما در صورتی که یکی از گره‌های شبکه دچار ایراد شود، بخش زیادی از شبکه مختل می‌شود. همچنین تغییر در یکی از گره‌های آن (جابجایی، اضافه کردن یا حذف کردن) معمولا روی کل شبکه تاثیرگذار است. شبکه حلقه نیز مقیاس‌پذیری بالایی ندارد. پردازنده‌های Intel Core i7 از این شبکه استفاده می‌کنند.
- مش (Mesh): شبکه مش به دلیل تقریبا هم‌اندازه بودن تمام ارتباطات آن به سادگی می‌تواند در یک چیپ پیاده‌سازی شود. همچنین مسیریابی در آن از چندین راه مختلف امکان‌پذیر است. در این شبکه گره‌های لبه‌ای به دلیل دور بودن از دیگر گره‌ها کارایی پایین‌تری نسبت به گره‌های میانی دارند. از مش در پردازنده ۱۰۰ هسته‌ای Tilera، پردازنده Teraflop اینتل و on-chip network prototypes استفاده می‌شود.
- توری (Torus): شبکه Torus مشکل عدم تقارن مش را حل کرده است و مکان قرارگیری گره‌ها تاثیری روی سرعت دسترسی آن به بخش‌های دیگر ندارد. همچنین مسیرهای بیشتری (نسبت به مش) بین گره‌های مختلف وجود دارد. هزینه پیاده‌سازی این شبکه بالاست و پیاده‌سازی آن داخل چیپ نیز به دلیل هم‌اندازه نبودن ارتباطات مختلف مشکل است. توروس معمولا در پیاده‌سازی سیستم‌های موازی بزرگ مانند IBM BlueGene استفاده می‌شود.

۲) در مش و توری d-بعدی درجه رئوس آن برابر 2d است و در نتیجه در شبکه ۲ بعدی هر راس درجه‌ای برابر ۴ (به جز رئوس کناری در مش) دارد. همچنین اتصال رئوس و یال‌ها در مش برابر d و در توری برابر 2d است. که به ترتیب در حالت ۲ بعدی برابر ۲ و ۴ است. این عوامل بر هزینه پیاده‌سازی مش و توری با n-گره تاثیر می‌گذارد.

---

۸. تسریع، بهره‌وری و مقیاس‌پذیری را تعریف نموده و سپس به سوالات زیر پاسخ دهید.

۱) در چه حالت هایی تسریع فوق خطی خواهیم داشت. با ذکر مثال توضیح دهید.

۲) انواع تسریع و عوامل موثر تسریع را نام ببرید.

۳) آیا با اضافه کردن n هسته در سیستم‌های چند هسته‌ای تسریع n برابر می‌شود؟ تمام حالات ممکن را بررسی نمایید.

۴) به ازای اندازه‌های مختلف مسئله تسریع و بهره‌وری چگونه تغییر می کنند.

۵) بهره‌وری و تسریع چه ارتباطی با هم دارند؟

۶) مقیاس‌پذیری چه رابطه‌ای با تسریع و بهره‌وری دارد؟

۷) انواع معیارهای مقیاس‌پذیری را بیان کنید.

۸) کاربرد و مزیت هر کدام از معیارهای مقیاس‌پذیری را شرح دهید.

تسریع: به معنی نسبت زمان اجرای برنامه در حالت سریال به حالت موازی است. یا $S = \frac{T_s}{T_p}$.

بهره‌وری: بازده یا بهره‌وری نسبت تسریع به تعداد هسته‌ی استفاده شده برای رسیدن به آن میزان از تسریع است. یا $E = \frac{S}{p}$.

مقیاس‌پذیری: سیستمی مقیاس‌پذیر است که اگر با افزایش تعداد نخ‌ها، بتوان اندازه مسئله را بزرگ‌تر کرد به گونه‌ای که بازده (بهره‌وری) ثابت بماند.

۱) در برخی مسائل حجم داده‌هایی که مورد نیاز است از حجم حافظه نهان (Cache) یک پردازنده بزرگ‌تر است. با شکستن مسئله و داده‌ها به قسمت‌های کوچک‌تر و سپردن قسمت‌های مختلف به پردازنده‌های مختلف، در صورتی که داده‌های توزیع شده در حافظه نهان پردازنده‌ها جا شود، می‌توانیم تسریع فوق خطی داشته باشیم.

یا به عنوان مثالی دیگر، در مسائلی که مربوط به جست‌وجو در یک فضای حالت است، اضافه کردن هسته‌های بیشتر باعث بیشتر شدن تعداد جست‌وجوگرها می‌شود. حال ممکن است به صورت تصادفی یکی از جست‌وجوگرها در نزدیکی حالت هدف قرار بگیرند. در چنین حالتی نیز می‌توانیم تسریع فوق‌خطی داشته باشیم.

۲) تسریع ممکن است زیرخطی، خطی یا فوق‌خطی باشد. عوامل موثر بر تسریع عبارتند از: الگوریتم مورد استفاده، نوع کامپایلر و تنظیمات بهینه‌سازی آن، سیستم عامل، فایل سیستم هارد دیسک و بار کاری سیستم در هنگام اجرای برنامه

۳) خیر. در صورتی که با n برابر کردن هسته‌ها، تسریع n برابر شود، تسریع خطی داریم. اگر کمتر از n برابر شود تسریع زیرخطی و اگر بیشتر شود تسریع فوق‌خطی داریم.

معمولا با اضافه کردن هسته‌های بیشتر، تسریع خطی نیست. زیرا سربارهای موازی‌سازی (مانند ساخت نخ‌ها، ارتباطات بین نخ‌ها و ...) عمدتا باعث کاهش سرعت برنامه موازی می‌شوند و این عامل سبب می‌شود تا تسریع زیرخطی داشته باشیم.

۴) با زیاد کردن تعداد هسته‌ها و ثابت ماندن اندازه مسئله (یا ثابت ماندن تعداد هسته‌ها و کاهش اندازه مسئله) سربار موازی‌سازی نسبت به میزان کار مفید موازی‌سازی افزایش می‌یابد و باعث کاهش تسریع و بازده می‌شود.

با افزایش اندازه مسئله، سربار موازی‌سازی به نسبت کار مفید کم‌تر می‌شود و تسریع و بازده افزایش می‌یابند.

۵) بهره‌وری در واقع معیاری است از به صرفه بودن تسریع انجام شده در برنامه. به این معنی که اگر با بهره‌وری پایین به تسریع بالایی دست یابیم، با هزینه و تعداد هسته‌های بسیار بالا به آن دست یافته‌ایم.

۶) مقیاس‌پذیری به این معنی است که با افزایش تعداد هسته‌ها (و تسریع برنامه) آیا بازده ثابت می‌ماند؟ اگر ثابت بماند برنامه قویا مقیاس‌پذیر است. اگر با افزایش اندازه مسئله ثابت بماند برنامه مقیاس‌پذیر است و در غیر این صورت مقیاس‌پذیر نیست.

۷) مقیاس‌پذیری را به دو صورت مقیاس‌پذیری قوی (Strong Scaling) و مقیاس‌پذیری ضعیف (Weak Sacling) می‌توان ارزیابی کرد.

مقیاس‌پذیری قوی که با قانون آمدال می‌توان آن را بررسی کرد، بیانگر آن است که با اندازه مسئله ثابت، حد بالای تسریع قابل دسترسی با S (کسری از برنامه که قابل موازی‌سازی نیست) مشخص می‌شود.

در مقیاس‌پذیری ضعیف که با قانون گوستافسون می‌توان آن را نشان داد، تسریع sacle شده، به صورت خطی با افزایش تعداد هسته‌ها بیشتر می‌شود و حد بالایی برای آن وجود ندارد. به این صورت که در این مقیاس‌پذیری تعداد هسته‌ها و اندازه مسئله هر دو افزایش می‌یابند و تسریع بررسی می‌شود.

۸) یکی از کاربردهای اندازه‌گیری مقیاس‌پذیری قوی و ضعیف در استفاده از کلاسترهای HPC است. زیرا با توجه به نتایج تست‌های مقیاس‌پذیری، می‌توان تعادل مناسبی بین اندازه مسئله و مقدار منابعی که در اختیار آن قرار می‌دهیم برقرارکرد.

همان‌طور که گفته شد، مزیت مقیاس‌پذیری قوی در آن است که پس از افزایش تعداد هسته‌ها، می‌توان تسریع آن را با توجه به منابع اضافه شده بررسی کرد. مقیاس‌پذیری ضعیف نیز با دخیل کردن اندازه مسئله در آن، میزان تسریع آن را بررسی می‌کند.

---

۹. قانون آمدال و قانون گوستافسون را توضیح دهید. سپس به سوالات زیر پاسخ دهید.

۱) بررسی کنید که چگونه به رابطه (۱) برای تسریع می‌رسیم.

$$S = \frac{1}{1 - a + \frac{a}{p}} \qquad \textbf{Equation 1}$$

۲) تفاوت‌های قانون گوستافسون و قانون آمدال را بیان کنید.

قانون آمدال بیشینه تسریع تئوری ممکن یک برنامه موازی (بدون در نظر گرفتن سربارهای موازی‌سازی، توزیع نامتوازن تسک‌ها بر روی هسته‌ها و ...) را با توجه به تعداد هسته‌ها و با اندازه مسئله ثابت بیان می‌کند.

قانون گوستافسون این تسریع را با توجه به تعداد هسته‌ها و اندازه مسئله بیان می‌کند. در نتیجه با آن می‌توان مقیاس‌پذیری یک سیستم را نیز بررسی کرد. در واقع اگر فرض کنیم f کسری از برنامه است که قابلیت موازی‌سازی ندارد و p تعداد هسته‌ها است، با توجه به قانون گوستافسون برای اینکه زمان اجرا ثابت بماند باید تسریع برابر باشد با:

$$S = p + (1 - p)f$$

۱) اگر کسری از برنامه که قابلیت موازی‌سازی دارد را a در نظر بگیریم و فرض کنیم زمان اجرای این قسمت از برنامه با اجرا روی p هسته، $1/p$ شود، آنگاه طبق تعریف تسریع داریم:

$$S = \frac{T_s}{T_p} = \frac{T_s}{(1-a)T_s + a\frac{T_s}{p}} = \frac{1}{(1-a) + \frac{a}{p}}$$

۲) در قانون آمدال، با فرض ثابت بودن اندازه مسئله، زمان اجرای آن را با اضافه کردن تعداد هسته‌های بیشتر محاسبه می‌کنیم. در حالی که در قانون گوستافسون بیان می‌شود که با گسترش سیستم‌های موازی، دیگر نیازی به ثابت بودن اندازه مسئله نیست و می‌توان مسائل با ابعاد بسیار بزرگ‌تر (که پیشتر حل آن‌ها غیرممکن بود) را نیز با اضافه کردن هسته‌های بیشتر، زمان اجرای آن را ثابت نگه داشت.

---

۱۰. آیا یک برنامه که دارای تسریع خطی است قویاً مقیاس‌پذیر است؟ پاسخ خود را شرح دهید.

بله. برنامه‌ای که تسریع خطی داشته باشد، تسریع آن $S = p$ است که در آن p برابر تعداد هسته‌ها است.

از طرفی طبق تعریف بهره‌وری داریم:

$$E = \frac{\mathrm{S}}{\mathrm{p}}$$

در نتیجه می‌بینیم که $E = \frac{\mathrm{p}}{\mathrm{p}} = 1$ است. پس این برنامه با افزایش تعداد هسته‌ها و بدون نیاز به تغییر اندازه مسئله، کارایی آن ثابت است و قویا مقیاس‌پذیر است.