سیستم‌های عامل - دکتر ابراهیمی‌مقدم

امیرحسین منصوری - ۹۹۲۴۳۰۶۹

تمرین سری سوم

سوال ۱

الف) *نادرست؛* فایل‌های باز، از منابعی هستند که بین threadهای یک پروسه مشترک هستند. بنابراین اگر یک پروسه در هر thread بتواند فایلی را باز کند، بقیه threadها نیز می‌توانند از آن فایل باز استفاده کنند.

ب) *درست؛* همه اعمال مربوط به user-level threadها در user-space انجام می‌شوند که در آن اجازه دسترسی به سخت‌افزار وجود ندارد و فقط از سیستم‌کال‌های سیستم‌عامل استفاده می‌شود. بنابراین نیازی به پشتیبانی سخت‌افزاری نیست.

ج) *نادرست؛* User threadها معمولاً سبک‌تر از Kernel threadها هستند؛ زیرا سربار Context-switch یا زمان‌بندی غیرانحصاری در آن‌ها وجود ندارد و نیازی به اجرای System callهای کند سیستم‌عامل برای مدیریت آن‌ها نیست.

د) *درست؛* فرآیند Context-switch در User threadها بسیار ساده‌تر است و معمولاً با صدا زدن یک تابع معمولی Runtime library و جابه‌جایی محتویات رجیسترها انجام می‌شود؛ اما در Kernel threadها ممکن است نیازمند یک Context-switch کامل باشیم که در آن، یک System-call کند سیستم‌عامل را صدا می‌زنیم و حتی ممکن است بعد از آن، Scheduler تصمیم بگیرد یک thread از یک پروسه دیگر را اجرا کند که در این صورت باید اطلاعات مربوط به پروسه‌ها نیز جابه‌جا شود و این کار بسیار کند است.

سوال ۲

سیستم *موازی*، قابلیت اجرای چند وظیفه به صورت همزمان را دارد (مثلا با به‌کارگیری چندین پردازنده). اما سیستم *همروند* صرفا می‌تواند چندین وظیفه را طوری مدیریت کند که با هم (و نه الزاما در آن واحد و همزمان) پیشروی کنند؛ به طور مثال، با انجام قسمت کوچکی از هر وظیفه و جابه‌جا شدن بین وظایف، می‌توان *توهمی* از همروندی را حتی در یک سیستم با یک پردازنده ایجاد کرد. درنتیجه سیستم همروند، الزاما موازی نیست.

سوال ۳

در موازی‌سازی با استراتژی Data parallelism، هر قسمت از مجموعهٔ دادهٔ خود را به یک هستهٔ پردازشی می‌دهیم و این هسته‌ها با انجام یک کار یکسان روی هر تکه از داده، در نهایت کل کار را به طور موازی انجام می‌دهند. در موازی‌سازی با استراتژی Task parallelism، یک داده یکسان به همه هسته‌ها داده می‌شود، اما هر هسته مشغول انجام یک کار متفاوت روی آن مجموعهٔ داده می‌شود؛ به عبارتی به جای داده، کل کار به چند قسمت کوچکتر تقسیم می‌شود و هر قسمت کوچک از کار، به یک هسته داده می‌شود.

به طور خلاصه، در Data parallelism، تقسیم داده را داریم، و در Task parallelism، تقسیم کار را داریم.

در مثال ایجاد thumbnail، چند thread داریم که هر کدام مشغول انجام کار ساخت thumbnail روی یک عکس از مجموعه‌ای از عکس‌ها هستند؛ بنابراین اینجا از استراتژی Data parallelism استفاده شده است.

در مثال ارسال و دریافت داده از شبکه، ما دو کار متفاوت داریم که بین دو thread مختلف پخش شده‌اند و روی داده‌های مختلفی کار می‌کنند؛ بنابراین اینجا از استراتژی Task parallelism استفاده شده است.

سوال ۴

منابعی مثل فایل‌های باز، حافظه، و کد برنامه بین threadها مشترک هستند؛ اما پشته و مقدارهای رجیستر، در هر thread منحصر به فرد است. با توجه به این که هر thread می‌تواند یک procedure متفاوت از کد پروسه را اجرا کند، در نتیجه هر ترد باید پشته منحصر به فرد خودش را داشته باشد، و همچنین بتواند مقادیر رجیستر متفاوت با بقیه threadها داشته باشد.

سوال ۵

- **مدل یک به یک:** به ازای هر user-level thread، یک kernel-level thread ساخته می‌شود. از مزیت‌های این روش این است که یک فراخوانی blocking دیگر کل پروسه را block نمی‌کند و بقیه threadها می‌توانند به اجرا شدن ادامه دهند. از معایب آن، سنگینی و کندی ساخت و مدیریت kernel-thread ها است.
- **مدل چند به یک:** در این حالت، همه user-level thread ها روی یک kernel-level thread اجرا می‌شوند. در این حالت مدیریت همه threadها در user-space انجام می‌شود و بنابراین ساخت و مدیریت آن‌ها سریعتر است. ایراد این روش این است که چون سیستم‌عامل این user-level threadها را نمی‌بیند، یک فراخوانی blocking همه threadها را به حالت انتظار می‌برد. همچنین این حالت اجازه parallelism را در حالتی که چندین هسته داشته باشیم نمی‌دهد.
- **حالت چند به چند:** این حالت، میان دو حالت قبلی است. اجرای user-level threadها روی تعداد برابر یا کمتری از kernel-level threadها پخش می‌شوند. این حالت اجازه parallelism را می‌دهد و هنگام اجرای یک فراخوانی blocking، همه threadها را مسدود نمی‌کند؛ اما پیاده‌سازی آن بسیار دشوار است.

سوال ۶

وقتی ٪۷۰ برنامه به صورت موازی اجرا می‌شود، ٪۳۰ دیگر آن به صورت سری اجرا می‌شود. طبق فرمول آمدال، با ۲ هسته پردازشی، مقدار speed-up برابر $s_1 = \frac{1}{0.3+\frac{1-0.3}{2}} \approx 154\%$ خواهد بود. با ۵ هسته، این مقدار برابر $s_2 = \frac{1}{0.3+\frac{1-0.3}{5}} \approx 227\%$ خواهد بود. بنابراین بهبود speed-up برابر خواهد بود با

$$\frac{s_2 - s_1}{s_1} = 47.7\%$$

سوال ۷

داده‌ای که هر thread یک کپی مخصوص خودش از آن دارد را Thread-local storage می‌گوییم. یک تفاوت TLS با متغیر محلی این است که متغیر محلی صرفاً در یک صدا زدن تابع دیده می‌شود، و بین توابع صدا زده شده یک thread مشترک نیست. اما TLS توسط همهٔ توابع صدا زده شده قابل دسترسی است. این موضوع یک شباهت به متغیرهای static محسوب می‌شود. یک تفاوت متغیر static با TLS نیز این است که متغیر static در همه threadها مشترک است، اما هر thread، یک ارجاع منحصر به فرد به TLS دارد و این ارجاع بین threadها مشترک نیست.

سوال ۸

با توجه به این که در کد هیچ‌جا مقدار z مثبت نمی‌شود، همواره فقط قسمت else شرط اجرا خواهد شد. قسمت‌هایی که متغیر a خوانده شده یا در آن نوشته شده را بررسی می‌کنیم. در T۱ و در خط ۴ و ۹، از a خوانده شده است؛ نام این خواندن‌ها را به ترتیب r_1 و r_2 می‌گذاریم. همچنین در خط ۱۱ نوشتن w_1 را داریم. همچنین در T۲ در خط ۱۵ خواندن r_3 و در خط ۱۶ نوشتن w_2 را داریم.

می‌توان دید که r_1 دقیقا قبل از r_2 انجام می‌شود و هر دو حاصل را در b ذخیره می‌کنند. بنابراین می‌توان r_1 را نادیده گرفت. حالت‌های مختلف ترتیب اجرای دیگر خواندن‌ها و نوشتن‌ها را بررسی می‌کنیم:

- $r_2 \to r_3 \to w_1 \to w_2$: ابتدا مقدارهای $b = 4$ و $c = 0$ نوشته می‌شوند. سپس چون w_2 دیرتر انجام می‌شود، درنهایت مقدار $a = 0$ نوشته می‌شود.
- $r_3 \to r_2 \to w_1 \to w_2$: ترتیب خواندن‌ها تفاوتی ندارد؛ بنابراین حاصل دقیق مشابه حالت قبلی است و $a = 0$ خواهد بود.
- $r_2 \to r_3 \to w_2 \to w_1$: اجرا مشابه حالت اول انجام می‌شود؛ اما چون w_1 دیرتر اجرا می‌شود، $a = 4$ خواهد بود.

- $r_3 \to r_2 \to w_2 \to w_1$: ترتیب خواندن‌ها تفاوتی ندارد؛ بنابراین حاصل دقیق مشابه حالت قبلی است و $a = 4$ خواهد بود.
- $r_2 \to w_1 \to r_3 \to w_2$: هنگام w_1، مقدار $a = 4$ نوشته می‌شود. سپس در r_3، مقدار $c = a - 2 = 2$ خوانده می‌شود. و در نهایت در w_3، $a = 2$ نوشته می‌شود.
- $r_3 \to w_2 \to r_2 \to w_1$: هنگام w_2، مقدار $a = 0$ نوشته می‌شود. سپس در r_2، مقدار $b = a = 0$ خوانده می‌شود. و در نهایت در w_3، $a = 0$ نوشته می‌شود.

در نتیجه سه حالت مختلف $a = 0, 2, 4$ می‌تواند به دست بیاید.