دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده مهندسی کامپیوتر

آزمایشگاه سیستم‌های عامل

آزمایش ۵: برنامه‌نویسی چند فرآیندی و رسم نمودار توزیع نرمال

صورت گزارش

مهلت تحویل: ۲۷ آبان ۱۴۰۳

مدرس: مینا یوسف‌نژاد

پیاده‌سازی محاسبات موازی با پردازش چندگانه

۱. برنامه‌ای بنویسید که با استفاده از چندین فرآیند، یک هیستوگرام از مقادیر تصادفی تولید شده ایجاد کند.
   - به هر فرآیند یک بخش از داده‌ها اختصاص دهید.
   - نتایج هر فرآیند را از طریق **pipe** به فرآیند والد ارسال کنید.
   - در نهایت، فرآیند والد باید هیستوگرام نهایی را با جمع‌آوری نتایج همه فرآیندها بسازد.
۲. با استفاده از دستور time، زمان اجرای برنامه را برای تعداد نمونه‌های ۱۰۰، ۱۰۰۰، ۱۰۰۰۰، ۱۰۰۰۰۰ محاسبه و ثبت کنید.
   - زمان کلی اجرای برنامه(real) ، زمان کاربر (user) و زمان سیستم (sys) را برای هر حالت یادداشت کنید.
۳. نتایج را در یک جدول ثبت کنید و مقایسه کنید. آیا با افزایش تعداد نمونه‌ها، زمان اجرا به صورت خطی افزایش می‌یابد؟ توضیح دهید.

(با کدهای موجود در github این تمرین را حل کنید (multiProcess2))

تحلیل تفاوت بین حالت سریال و موازی

۱. یک نسخه از برنامه هیستوگرام (مانند برنامه‌ای که در تمرین قبل نوشتید) را به صورت **سریال** پیاده‌سازی کنید. در این نسخه، همه محاسبات باید در یک فرآیند انجام شوند.
۲. دوباره، با استفاده از timeزمان اجرای نسخه سریال را برای تعداد نمونه‌های مختلف ثبت کنید و در یک جدول وارد کنید.
۳. حالا زمان اجرای نسخه موازی (تمرین ۱) را با نسخه سریال مقایسه کنید. برای هر اندازه از نمونه‌ها، نسبت سرعت اجرای موازی به سریال را محاسبه کنید و نتیجه را تفسیر کنید.

(با کدهای موجود در github این تمرین را حل کنید (singleProcess))

بهبود کارایی برنامه با افزایش تعداد فرآیندها

۱. در برنامه هیستوگرام موازی خود، تعداد فرآیندها (PROCESS_COUNT) را افزایش دهید و نتایج زمان اجرای آن را برای تعداد فرآیندهای مختلف (۲، ۴، ۸ و ۱۶ فرآیند) ثبت کنید.
۲. نمودار هیستوگرام آن‌ها را رسم کنید.
۳. بر اساس نمودار، تحلیل کنید که با افزایش تعداد فرآیندها، کارایی برنامه چگونه تغییر می‌کند. آیا افزایش تعداد فرآیندها به طور مداوم باعث بهبود عملکرد می‌شود؟ چرا؟

بررسی تأثیر ارتباطات بین‌پردازشی

۱. در برنامه موازی خود، به جای استفاده از **pipe**برای ارتباط بین فرآیندها، از **shared memory**(حافظه اشتراکی) استفاده کنید.

- داده‌ها را مستقیماً در حافظه اشتراکی ذخیره کنید و در پایان محاسبات، فرآیند والد هیستوگرام نهایی را از حافظه اشتراکی استخراج کند.

۲. زمان اجرای برنامه با استفاده از حافظه اشتراکی را ثبت کنید و با زمان اجرای نسخه‌ای که از pipe استفاده می‌کند مقایسه کنید.

۳. تحلیل کنید که آیا استفاده از حافظه اشتراکی باعث بهبود عملکرد شده است؟ در چه شرایطی ارتباط با حافظه اشتراکی بهتر از pipe عمل می‌کند؟

رسم هیستوگرام با استفاده از داده‌های ذخیره شده

۱. در برنامه خود، کدی اضافه کنید که داده‌های هیستوگرام را به صورت فایل خروجی (مانند histogram_data.txt) ذخیره کند.

۲. از ابزارهایی مانند **gnuplot** یا **Python (matplotlib)** برای رسم هیستوگرام استفاده کنید. فایل خروجی را در برنامه رسم بارگذاری کنید و نتایج را به صورت نمودار نمایش دهید.

۳. برای رسم هیستوگرام از:Python

- برنامه‌ای کوتاه در Python بنویسید که فایل histogram_data.txt را بخواند و با استفاده از matplotlib آن را رسم کند.

محاسبه سرعت‌پذیری و کارایی

۱. سرعت‌پذیری (Speedup) را برای نسخه موازی و سریال برنامه محاسبه کنید:

$$\frac{{}_{serial}T}{{}_{parallel}T} = Speedup$$

- T_serial زمان اجرای نسخه سریال برنامه و T_parallel زمان اجرای نسخه موازی است.

۲. کارایی (Efficiency) را نیز با توجه به تعداد فرآیندها محاسبه کنید:

$$\frac{Speedup}{\text{Number of Processes}} = Efficiency$$

۳. نتایج را در جدول ثبت کرده و تحلیل کنید که چگونه سرعت‌پذیری و کارایی با افزایش تعداد فرآیندها تغییر می‌کنند.