الگوریتم‌های چندریسه‌ای (Multithreaded Algorithms)

- الگوریتم‌های ترتیبی
- الگوریتم‌های موازی
- مدل‌های چند پردازنده‌ای
  - چندهسته‌اس
  - مدل حافظه‌ی مشترک
  - مدل توزیع‌شده

چند ریسه‌ای

- ریسه (thread) چیست؟
- برنامه‌نویسی چندریسه‌ای ایستا
  معمولاً ریسه‌ها در طول اجرا وجود دارد
- برنامه‌نویسی چندریسه‌ای پویا (dynamic multithreaded programming)
  ریسه‌ها به‌صورت پویا تولید و حذف می‌شوند.

مدل چندریسه‌ای پویا

اعمال spwan، sync و parallel

مثال: محاسبه‌ی اعداد فیبوناچی

$$
\begin{aligned}
F_{۰} &= ۰ \\
F_{۱} &= ۱ \\
F_i &= F_{i-۱} + F_{i-۲}, i > ۱
\end{aligned}
$$

راه‌حل ترتیبی

- راه‌حل پیشین $F_i - ۱$ عمل جمع دارد و از $\Theta(\Phi^n)$ است که $\Phi = (۱ + \sqrt{۵})/۲$.
- می‌توان در زمان خطی هم حل کرد.

• می‌دانیم

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

برای $n \geq 2$ و طبق تعریف اصلی رابطه‌های زیر برقرارند.

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n .$$

پس برای به‌دست آوردن F_n باید ماتریس $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ را n بار در خودش ضرب کنیم و این کار را می‌توان در $\Theta(\lg n)$ و بدون خطای محاسباتی دقیقاً محاسبه کرد.

راه‌حل چندریسه‌ای

مفاهیم

- strand، **قطعه کدی که در دستورهای موازی ندارد**
- **معیارها:** span **(زمان پایان)** و work **(کار)**
- **مسیر بحرانی**
- **با** P **پردازنده، زمان اجرا** T_P

روشن است که

- $T_P \geq T_1/P$ **(قانون کار)**
- $T_P \geq T_\infty$ **(قانون زمان پایان)**
- **تسریع** (speedup) $T_1/T_P \leq P$.
- **اگر** $T_1/T_P = \Theta(P)$، **تسریع خطی**
- **اگر** $T_1/T_P = P$، **تسریع کامل**

- **$T_۱/T_\infty$ را موازات** parallelism
  - **-- برابر میانگین کار که می‌تواند در هر مرحله‌ی مسیر بحرانی انجام شود.**
  - **-- کران بالای بیش‌ترین میزان تسریع ممکن با هر تعداد پردازنده**
  - **-- حد بالای حالتی که ممکن است تسریع خطی داشت. اگر تعداد پردازنده‌ها از موازات بیش‌تر شود نمی‌توان انتظار تسریع خطی را داشت.**

    یعنی اگر $P > T_۱/T_\infty$، در آن صورت، $T_۱/T_P \leq T_۱/T_\infty < P$

    یا اگر $P >> T_۱/T_\infty$، در آن صورت، $T_۱/T_P << P$
- **در مورد اعداد فیبوناچی، $T_۱ = ۱۷$ و $T_\infty = ۸$ پس موازات برابر $۲.۱۲۵ = ۱۷/۸$ است.**

- **$(T_1/T_\infty)/P$ را** slackness **می‌گوییم که عددی بین ۱ تا صفر است.**
  - **اگر** $1 <$ slckness **در آن‌صورت،** $T_1/T_P < T_1/T_\infty < P$
  - **اگر** $1 <$ slckness **و به ۰ نزدیک بشود، در آن‌صورت، تسریع از خطی دورتر می‌شود.**
  - **اگر** $1 >$ slckness**، تسریع به خطی نزدیک‌تر می‌شود.**

زمان‌بندی

- زمان‌بندی برخط
- زمان‌بندی حریصانه
- زمان‌بندی مرکزی
- بهترین زمان پایان با P پردازنده $T_P = T_1/P$ و از طرفی $T_P \leq T_\infty$

قضیه: در یک کامپیوتر موازی ایده‌آل با زمان‌بندی حریصانه داریم

$$T_P \leq T_1/P + T_\infty$$

اثبات: جمع تعداد مرحله‌های کامل و ناکامل

تعداد مرحله‌های کامل: $\lfloor T_۱/P \rfloor$

تعداد مرحله‌های نا کامل: حداکثر برابر است با T_∞

نتیجه: $T_P \leq ۲T_P^*$ **در زمان‌بندی حریصانه بر روی کامپیوتر ایده‌آل موازی**

$$\begin{aligned} T_P &\leq T_۱/P + T_\infty \\ &\leq ۲.\max\{T_۱/P, T_\infty\} \\ &\leq ۲T_P^* \end{aligned}$$