MI-SPOL-11

Výkonnostní měřítka paralelních algoritmů, PRAM model, APRAM model, škálovatelnost.

Paralelní počítač: skupina propojených výpočetních prvků, které komunikují a spolupracují, aby rychleji vyřešily velké a náročné výpočetní úlohy

- **Jádro:** Může soušasně provádět více vláken
- **Vícejádrové CPU:** několik jader uvnitř CPU
- **GPUs:** Tisíce menších synchronně pracujících jader
- **Výpočetní SMP (symmetric multiprocessor) uzly:** několik (desítky) vícejádrových CPU se sdílenou pamětí
- **Výpočetní klastr:** stovky až desetitisíce výpočetních SMP uzlů
- **Masivně paralelní superpočítače:** deseti- až statisíce výpočetních uzlů propojených několika speciálními rychlými sítěmi
- **Cloud computing:** ICT infrastruktura, datová centra...

Modely paralelních systémů:

- **Modely výpočetního stroje:** HW, ISA, OS
- **Modely architektury:** toky, dat/instrukcí, organizace paměti, propojovací sítě
  - **SIMD** - single instruction, multiple data - všechny uzly přijímají společný proud instrukcí, které buď provedou, nebo ignorují *(GPU, vektorové koprocesory)*
  - **MIMD** - každý uzel samostatný počítač s instrukční a datovou pamětí *(vícejádrová CPU, klastry...)*
  - **Organizace paměti:**
    - sdílená (UMA)
    - distribuovaná (NUMA) - HW/SW komunikace
    - virtuálně sdílená (CC-NUMA)
- **Výpočetní modely:** analytický model architektury pro navrhování a verifikaci paralelních programů
- **Programovací modely:** sémantika paralelních vyšších jazyků a prostředí, model přístupu do sdílené paměti

Výpočetní model Paralelní RAM (PRAM)

Množina p procesorů RAM $P_1, P_2, ..., P_p$.

Každý P_i má **vlastní lokální (soukromou) paměť** a zná svůj index i.

Pole m **sdílených paměťových buněk** $M[1], M[2], ..., M[m]$
Každý P_i má přístup do každé $M[j]$ v čase $O(1)$ -- řešení problémů musí být **explicitně ošetřeno**

Vstup PRAM algoritmu: n položek v n buňkách sdílené paměti

Výstup PRAM algoritmu: n' položek v n' buňkách sdílené paměti

PRAM procesory provádějí **3 typy instrukcí:**

- **Čtení** buňky sdílené paměti (`READ` , `R`)
- **Lokální operace** (`LOCAL` , `L`)
- **Zápis** do buňky sdílené paměti (`WRITE` , `W`)

Jediný způsob **komunikace procesorů:** `READ` / `WRITE` do sdílené buňky

Jednotkový model: jednotkový čas každé operace `R` / `L` / `W`
Globální model: `L` trvá čas 1 a `R` / `W` trvají konstantní čas $d > 1$

Ošetření konfliktů při přístupech do sdílené paměti PRAM

- **Exclusive Read Exclusive Write (EREW):** žádné 2 procesory nesmějí READ nebo WRITE do téže paměťové buňky najednou
- **Concurrent Read Exclusive Write (CREW):** současná čtení téže pamě´tové buňky povolena, zápis ne
- **Concurrent Read Concurrent Write (CRCW:):** Povolena současná čtení i zápisy
  - **Priority-CRCW-PRAM:** Procesory mají pevné priority, zápis povolen tomu s nejvyšší
  - **Arbitrary-CRCW-PRAM:** Zápis povolen náhodnému procesoru (algoritmus nesmí na výběr spoléhat)
  - **Common-CRCW-PRAM:** Všem žádajícím proecsorům je povoleno zapsat $\Leftrightarrow$ všechny zapisované hodnoty jsou stejné -- algoritmus toto musí zajistit

Asynchronní PRAM (APRAM)

Procesory pracují **asynchronně** -- neexistují centrální hodiny

Operace `READ`, `WRITE` a `LOCAL`

Nutná **explicitní synchronizace** -- bariéry

Doba přístupu do sdílené paměti **není jednotková**

APRAM výpočet: posloupnost globálních fází, ve kterých procesory pracují asynchronně, oddělených bariérovou synchronizací

Dva procesory nemohou přistupovat do téže buňky sdílené paměti v téže globální fázi, pokud jeden z nich zapisuje.

Výkonnostní parametry APRAM

- lokální operace: 1
- globální `READ` nebo `WRITE`: d
- k po sobě jdoucích globálních `READ` nebo `WRITE`: $d + k - 1$
- bariérová synchronizace: $B(p)$

d a B neklesající funkce p

Implementace bariéry

- **Centrální čítač:** inicializovaný na 0, příchozí proces inkrementuje, deaktivuje se a čeká na aktivaci. $B(p) = \Theta(dp)$
- **Binární redukční strom:** Proces dorazí k bariéře, čeká až skončí redukce v jeho podstromu a pošle signál rodiči. Kořen počká na redukci z obou podstromů a bariéra je prolomena. $B(p) = \Theta(d \log p)$

Měřítka složitosti sekvenčních algoritmů

- $T_A^K(n)$: **časová složitost/doba výpočtu** sekvenčního algoritmu A, který řeší problém K na vstupních datech velikosti n
- $SL^K(n)$: **spodní mez** sekvenční časové složitosti řešení problému K (nejhorší časová složitost

nejlepšího možného sekvenčního algoritmu pro řešení K)
- $SU^K(n)$: **horní mez** časové složitosti pro řešení problému K (nejhorší časová složitost nejrychlejšího známého sekvenčního algoritmu pro K)

A je **asymptoticky optimální** sekvenční algoritmus pro $K \Leftrightarrow T_A(n) = \Theta(SU^K(n)) = \Theta(SL^K(n))$

A je **nejlepší známý** sekvenční algoritmus pro řešení K, pokud $T_A(n) = \Theta(SU^K(n)) = \omega(SL^K(n))$

Paralelní měřítka

Paralelní časová složitost: $T(n,p)$: čas, který uplynul od začátku paralelního výpočtu do okamžiku, kdy poslední procesor skončil výpočet
* Závisí na architektuře
* Měřen čítáním výpočetních kroků a komunikačních kroků

Paralelní zrychlení: $S(n,p) = \frac{SU(n)}{T(n,p)} \leq p$
Paraelní zrychlení je **lineární** právě když $S(n,p) = \Theta(p)$ -- nejvyšší cíl paralelního programování
Superlineární zrychlení: více než lineární. Příčiny: jednomu CPU nestačila paměť, multi-CPU ano // anomálie při prohledávání kombinatorického stavového prostoru

Spodní mez na paralelní čas: $L^K(n,p) = \frac{SL^K(n)}{p}$

Paralelní cena: $C(n,p) = p \times T(n,p)$. Platí, že $C(n,p) = \Omega(SU(n))$. Cena ukazuje, jestli není moc velký počet vláken při výpočtu neaktivní.
Paralelní algoritmus má **optimální cenu**, pokud $C(n,p) = O(SU(n))$ (potom tedy $C(n,p) = \Theta(SU(n))$)

Paralelní efektivnost: $E(n,p) = \frac{SU(n)}{C(n,p)} \leq 1$. Udává *relativní vytížení* výpočetních zdrojů během výpočtu. Představuje *zrychlení na jádro*: $E(n,p) = \frac{SU(n)}{C(n,p)} = \frac{S(n,p) \times T(n,p)}{p \times T(n,p)} = \frac{S(n,p)}{p} \leq 1$

Paralelní algoritmus má **konstantní efektivnost**, pokud $E(n,p) \geq E_0$ pro danou $0 < E_0 < 1$

Paralelní algoritmus je **cenově-optimální** $\Leftrightarrow$ má **lineární zrychlení** $\Leftrightarrow$ má **konstantní efektivnost**

Škálovatelnost: schopnost paralelního algoritmu držet paralelní optimalitu, pokud p a n rostou nebo klesají.
Typy škálovatelnosti:

- **Silná:** schopnost paralelního algoritmu pro fixní n dosáhnout lineárního zrychlení s rostoucím p *(měřítko rychlosti poklesu efektivnosti, pokud roste p při pevném n)*
- **Slabá:** definuje, jak se mění paralelní čas s p pro fixní n/p *(měřítko růstu n takového, že při rostoucím p zůstává efektivnost stejná)*

Amdahlův zákon

Každý sekvenční algoritmus A s časem $T_A(n)$ nad daty velikosti n se skládá z:

- **inherentně sekvenčního podílu** f_s, který může provést pouze jedno vlákno
- **paralelizovatelného podílu** $1 - f_s$

Nechť je sekvenční algoritmus A paralelizován pro pevné n pomocí $p > 1$ vláken. Pro zrychlení A při p vláknech platí ideálně **Amdahlův zákon saturace paralelizace**:

$$S(n,p) = \left(\frac{\text{čas bez vylepšení}}{\text{čas s vylepšením}}\right) = \frac{T_A(n)}{f_s \cdot T_A(n) + \frac{1-f_s}{p} \cdot T_A(n)} = \frac{1}{f_s + \frac{1-f_s}{p}} \leq \frac{1}{f_s}$$

$\Rightarrow$ Nezávisle na tom, kolik je spuštěno vláken, zrychlení nemůže přesáhnout $\frac{1}{f_s}$
$\Rightarrow$ Po jisté hranici nemá smysl přidávat vlákna, protože pro ně není práce

Gustafsonův zákon

Amdahlův zákon říká, že problém fixní velikosti poskytuje omezené množství paralelismu.

Gustafsonův zákon říká, že s rostoucím p se má úměrně navyšovat velikost problému n. Inherentně sekvenční část pak trvá vždy **konstantní čas** t_{seq} nezávisle na p. Inherentně paralelní část $t_{par}(n,p)$ bude **lineárně škálovat** s p v čase.
Potom:

$$S(n,p) = \frac{t_{seq} + t_{par}(n,1)}{t_{seq} + t_{par}(n,p)}$$

Pokud je paralelní část perfektně paralelizovaná, pak:

$$t_{par}(n,1) = SU(n) - t_{seq}$$

$$t_{par}(n,p) = \frac{SU(n) - t_{seq}}{p}$$

Potom

$$S(n,p) = \frac{t_{seq} + SU(n) - t_{seq}}{t_{seq} + \frac{SU(n) - t_{seq}}{p}} = \frac{\frac{t_{seq}}{SU(n) - t_{seq}} + 1}{\frac{t_{seq}}{SU(n) - t_{seq}} + \frac{1}{p}}$$

Tudíž

$$\lim_{n \to \infty} S(n,p) = p$$

pro jakoukoliv monotonně rostoucí funkci $SU(n)$.

Izoefektivní funkce

Dána konstanta $0 < E_0 < 1$. **Izoefektivní funkce**

- $\psi_1(p)$ je **asymptoticky minimální** funkce taková, že $\forall n_p = \Omega(\psi_1(p)) : E(n_p, p) \geq E_0$
- $\psi_2(n)$ je **asymptoticky maximální** funkce taková, že $\forall p_n = O(\psi_2(n)) : E(n, p_n) \geq E_0$

Funkce jsou počítány osamostatněním dané veličiny ze vztahu pro efektivitu.

$\psi_1(p)$ -- asymptoticky jak velké problémy můžu řešit s daným počtem procesorů a danou efektivitou?

$\psi_2(n)$ -- asymptoticky kolik max procesorů potřebuji pro danou velikost problému a danou efektivitu?