1 Paralelní algoritmy

Kritéria složitostí paralelních výpočtů, Optimální algoritmy, Škálovatelné systémy a algoritmy, Gustafsonův zákon

teorie složitosti se zabývá asymptotickými tendencemi funkcí popisujících složitosti. Zkoumat složitosti až na konstanty (skryté).

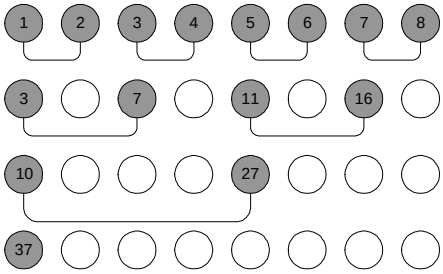
- **Spodní mez** je nejmenší možná dosažitelná složitost, kterou nelze překonat
- **Horní mez** je složitost nejlepšího známého algoritmu
  Def. nechť N^+ označuje množinu přirozených čísel a R^+ označuje množinu reálných čísel. nechť *f,g* jsou 2 funkce $N^+ \rightarrow R^+$
- O funkci *f(n)* řekneme, že je **řádu nejvýše** *g(n), f(n) = O(g(n))* iff podíl *f(n)/g(n)* nepřesáhne od jistého určitého *n* určitou kladnou konstantu. $\exists c \in R^+, \exists n_0 \in N^+, \forall n \geq n_0 (f(n) \leq c * g(n))$
- O funkci *f(n)* řekneme, že je **řádu nejméně** *g(n), f(n) = Ω(g(n))* iff podíl *f(n)/g(n)* neklesne od jistého určitého *n* určitou kladnou konstantu. $\exists c \in R^+, \exists n_0 \in N^+, \forall n \geq n_0 (f(n) \geq c * g(n))$
- O funkci *f(n)* řekneme, že je **stejného řádu jako** *g(n), f(n) = Θ(g(n))* iff jsou stejné až na multiplikativní konstantu, čily *), f(n) = O(g(n))* a *f(n) = Ω(g(n))*
- O funkci *f(n)* řekneme, že je **striktně nižšího řádu než** *g(n), f(n) = o(g(n))* iff podíl *f(n)/g(n)* je od jistého určitého *n* menší než jakákoliv konstanta $\exists c \in R^+, \exists n_0 \in N^+, \forall n \geq n_0 (f(n) < c * g(n))$
- O funkci *f(n)* řekneme, že je **striktně vyššího řádu než** *g(n), f(n) = o(g(n))* iff podíl *f(n)/g(n)* je od jistého určitého *n* větší než jakákoliv konstanta $\exists c \in R^+, \exists n_0 \in N^+, \forall n \geq n_0 (f(n) > c * g(n))$

Sekvenční algoritmy

časová složitost - sekvenční algoritmus = počet instrukcí, výpočetních kroků, operací.
spodní mez - K = problém, $SL^K(n)$ nejhorší časová složitost nejlepšího možného. nelze niky vyřešit lépe.
triviální spodní mez - odvozuje se od velikosti množiny vstupních, výstupních dat.
horní mez (sekvenční složitost) - $SU^K(n)$ - nejhorší časová složitost nejhoršího známého sekvenčního algoritmu
optimální sekvenční algoritmus SL = ST = SU. Pokud by byl znám optimální paralelní algoritmus, pak jeho triviální sekvenční simulací bychom dosáhli optimální sekvenční algoritmus Proto se pojem optimální paralelní algoritmus vztahuje k nejlepšímu známému sekvenčnímu algoritmu. T(n,p) = Θ (SU(n) / p)
nejlepší známý i když zatím neoptimální SL < ST = SU

Paralelní algoritmy
navíc dimenze počtu procesorů.
paralelní čas - T(n,p) je čas který uplyne od začátku paralelního výpočtu do okamžiku kdy poslední procesor skončí výpočet (měřen počtem výpočetních kroků, komunikačních kroků = architektura
spodní mez paralelního času je nemenší možný čas, za který může p procesoru vyřešit daný problém L(n,p) = SL(n) / p.
paralelní cena se nazývá součin procesory x čas. C(n,p) = p x T(n,p),
cenově optimální if C(n,p) = O (SU(n))
paralelní práce - W(n,p) přesnější, odpovídá počtu paralelně provedených operací. To = T(n,p)
W(n,p) = p1 + p2 + ... pi, kde pi je počet procesorů aktivně pracujících v kroku i (1..To)
pracovně optimální - W(n,p) = O (SU (n))
paralelní zrychlení S(n,p) kolikrát je paralelní algoritmus rychlejší než nejlepší znýmý sekvenční algoritmus S(n,p) = SU(n) / T (n,p)
paralelní efektivnost E(n,p) poměr sekvenční a paralelní operační složitosti E(n,p)=SU(n)/C(n,p). jinak řečeno je to zrychlení na 1 procesor.
paralelní režie - H(n,p) o kolik provede více operací než sekvenční H(n,p)=C(n,p)-SU(n)

paralelní zrychlení
lineární zrychlení - pokud počet procesorů stoupne k-krát, čas se zkrátí k-krát.
superlineární zrychlení - čas se zkrátí více než k-krát. Způsobeno velká množina dat a nevejde se do HP. Rozděl a panuj.



SU = SL
T(n,n) log n
C(n,n) n log n
W(n,n) = n + n/2 + n/4 + n/8 = 2n-2
S(n,n) n / log n
E(n,n) 1 / log n

Škálovatelnost - Brentnův simulační princip - simulace nemůže mít řádově horší práci ani cenu. Uvažujme problém K se vstupními daty o velikosti n. který lze řešit v t paralelních krocích za předpokladu, ž je k dispozici neomezený počet procesorů a komunikační režii lze ignorovat. nechť mi je počet operací provedených v paralelním kroku i. pak W(n,p)= sum mi. a stačí použít p = max mi procesorů, čímž dostaneme C(n,p) = pt = mi*t. Uvažujme p' procesorový počítač M s p'<p procesory. Jestliže lze u M též ignorovat komunikační režii mezi operacemi při řešení K na M, lze tentýž výpočet na M provést v T'(n,p') paralelních krocích.

Amdahalův zákon - pro dané n má smysl zvyšovat p do Ψ2(nú. Poté dochází k saturaci, další procesory nemohou přispět k rychlejšímu řešení, zrychlení stagnuje a efektivnost klesá.
Gustavsonův zákon - Pro velké instance problémů se sekvenční složka může stát téměř relativně zanedbatelná. Jestliže vrozeně paralelní část problému roste lineárně s p, takový algoritmus dosahuje lineárního zrychlení. S(n,p) = fs + p*fp = fs + p(1 - p) = p + fs(1 - fs) = p(1 - fs + fs/p)
lim p->oo S(n,p) = p*(1-fs) = Θ(p)
fs, fp relativní podíl vrozeně sekvenční, paralelní části.

PRAM modely a jejich vzájemná simulace
PRAM vychází z RAM (Random Access Machine)

- výpočetní jednotka s programem
- komunikuje přes vystupní / vstupní pásku.
- neomezený počet mem (číslo neomezené)
- výpočet začne Start a končí HALT
- instrukce trvají jednotkový čas
- **časová složitotst** = počet instrukcí
- **prostorová složitost** = used mem

PRAM

- neomezený počet RAM
- mem je tvořena neomezeným počtem sdílených buněk Mx, konflikt R-R, W-W explicitně
- Pi má lokální paměť
- Pi může přistupovat k Mx v jednotkovém čase
- vstup PRAM v M v n buňkach
- výstup opět v M
- instukce 3 cykly **synchronně**
  - čti z M do lokálu
  - proveď výpočet na lokalem (registry)
  - zapíš z regstru do M
- výměna dat pouze přes M
- P1 aktivní na start, také ukončuje (hierarchie)
- **paralelní časová složitost** = čas výpočtu P1
- **prostorová** = počet used mem

Mem konflikt

- EREW Exclusive Read Exclusive Write
- CREW Concurrent Read Exclusive Write
- CRCW Cincurrent Read Concurrent Write
  - Priority - podle přiřazené priority
  - Arbitary (náhodnýú - ukončit zápis může náhodně kdokoliv
  - Commnon (shodný) - zapíše se pokud všichni zapisují stejnou hodnotu

- zda je v poli x?
EREW PRAM T(n,p) = O(log p + n/p)

a) P1 rozešle x P2, .. Pn binárním distribucí (log p)
b) Každý provede lokální hledání v n/p
c) paralelní redukcí oznámí P1 v log p

CREW podobně jako EREW ale krok a) v 1 taktu
COMMON CRCW a) i b) 1 takt = T(n,p) = n/p

PRAM podmodel A je **výpočetně silnější** než podmodel B, A>=B. if jakákoliv algo. napsaný pro PRAM B poběží na stejně velkém PRAM A beze změny a s týmž paralelním časem.
Prioritní CRCW >= Nahorny CRCW >= Shodny CRCW >= CERW >= EREW

Simulace velkého PRAM na malém PRAM.p'<p Uvažujme algoritmus A, který běží na p-procesorovém PRAM v t krocích. Pak lze A simulovat na p'-procesorovém PRAm v t' = O (tp / p') krocích. za předpokladu, že velikost sdílené paměti je stejná.

Simulace silnějšího PRAM na slabším
Uvažujme Prioritní CRCW PRAM Jeden krok p-procesorového Prioritního PRAm s m mem (sdílenáú lze simulovat na p-procesorech EREW PRAM s mp mem v log p krocích.

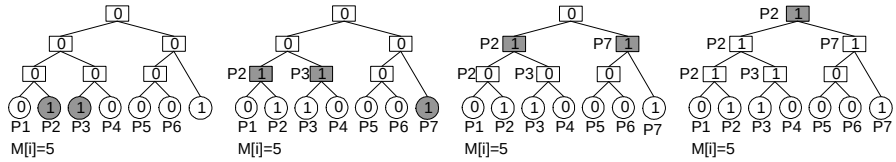
a) 1 proc. Prioritního simuluje 1 EREW
b) každá buňka mem v prioritním je simulována polem p buněk sdílené mem na EREW. Prazdné a organizovány jako uzly binárního stromu.

WRITE

1) všechny buňky mají příznak prázdný = 0
2) if Pk chce zapisovat do Mi, nastaví příznak na aktivní.
3) každý levý nastaví příznak do rodičovské buňky stormu.
4) každý pravý zkotroluje rodičovskou buňku, if prázdná nastaví své ID, jinak nečinný
5) oakuj log p krát
6) procesor v kořenu vyhrál

READ

1) paralelně provedou stejné průchody, aby určily vítěze skupin. Ti , 1<= i <= m.
2) vítězové přečtou hodnoty z buněk
3) během zpětného průchodu stromem Ti si procesory, které prohráli kopírují načtenou hodnotu a nastavují příznak na prázdná = 0.



a) p v Prioritnín CRCW je simulován p v EREW
b) každá M v CRCW je simulována M v EREW
c) EREW používá pomocné pole A s p buňkami
d) Chce li Pk přístup do Mi zapíše dvojici (i,k) do buňky Ak
e) všech p simulujících procesorů paralelně setřídí A v lexikografickém pořadí (log p)
f) každý Pk nastaví příznak s=0 if je li požadavek na mem = 0, nebo je li shodná mem s předchozí buňkou Ak-1, jinak s=1

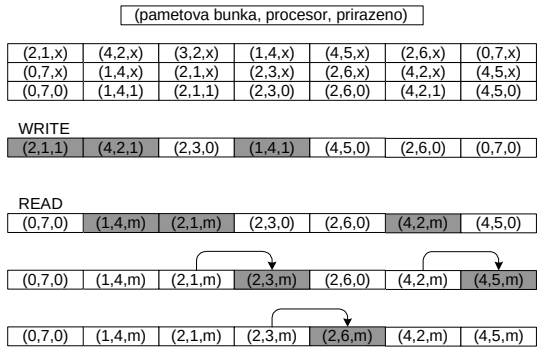
WRITE

g) každý Pj přečte trojici (i,k,s) ze své Aj a zapíše ji na správné místo Ak
h) každý Pk si přečte trojici (i,k,s) ze své buňky a pokud s=1 vyhrál

READ

g) každý Pk přečte trojici (i,k,s) ze své Ak
h) pokud s=1 požadavek na čtení výtězný, a Pk přečte hodnotu
i) binárním kopírováním se zkopíruje dál
j) pak Pj přečte svoji Aj přehodí ji na správné místo k procesoru Pk dle buňky Aj (i,k,s)
k) každý si přečte svoji hodnotu

P1 M2, P2 M4, P3 M2, P4 M1, P5 M4, P6 M2, P7 nic



Asynchronní PRAM (APRAM)

- pracují asynchronně neexistují centrální hodiny
- nutná explicitní synchronizace (bariéry)
- přístup do mem sdílené **není** jednotkový
- pokud procesor v dané glogální fázi zapisuje do mem pak žádný jiný nesmí v téže fázi přistupovat do stejné buňky
- lokální operace 1
- globální čtení zápis d
- k po sobě jdoucích čtení zápisů globálních d + k + 1 (vlastnos současných sběrnicových systémů, podporována schopnost zaznamenat více požadavků na přidělení sběrnice)
- bariérová synchronizace b(p)

simulace PRAM na APRAM

PRAM	**APRAM**
globální čtení	globální čtení
	bariéra B1
lokální vypočet	lokální vypočet
globální zápis	globální zápis
	bariéra B2

Snadno a těžko paralelizovatelné problémy (třída NC a P-úplné problémy)

Základní paralelní NC algoritmy na PRAM a běžných sítích (paralelní redukce, prefix výpočet, přeskoky ukazatelů, konstrukce Eulerovy cesty

Paralelní algoritmy pro třídění, třídící sítě

Paralelní algoritmy pro lineární alegebru