IB109 Návrh a implementace paralelních systémů

Analytický model paralelních programů

Jiří Barnat

Analytický model paralelních programů

Vyhodnocování sekvenčních algoritmů

- Časová a prostorová složitost
- Funkce velikosti vstupu

Vyhodnocování paralelních algoritmů

- Paralelní systém = algoritmus + platforma
- Složitost
- Přínos paralelismu
- Přenositelnost
- ...

V této kapitole

- Jak posuzovat výkon paralelních algoritmů a systémů

Režie (overhead) paralelních programů

Režie (overhead) paralelních programů

N-násobné zrychlení

- S použitím n-násobných HW zdrojů, lze očekávat n-násobné zrychlení výpočtu.
- Nastává zřídka z důvodů režií paralelního řešení.
- Pokud nastane, je jeho důvodem často neoptimální řešení sekvenčního algoritmu.

Důvody

- Režie – interakce, prostoje, složitost paralelního algoritmu
- Nerovnoměrné zvyšování HW zdrojů
  - Zvýšení počtu procesorů nepomůže, pokud aplikace není závislá na čistém výpočetním výkonu

Režie (overhead) paralelních programů

Komunikace

- Cena samotné komunikace.
- Příprava dat k odeslání, zpracování přijatých dat.

Prostoje (Lelkování, Idling)

- Nerovnoměrná zátěž na jednotlivá výpočetní jádra.
- Čekání na zdroje či data.
- Nutnost synchronizace asynchronních výpočtů.

Větší výpočetní složitost paralelního algoritmu

- Sekvenční algoritmus nejde paralelizovat (DFS postorder).
- Typicky existuje vícero paralelních algoritmů, je nutné charakterizovat, čím se platí za paralelismus.

Metriky výkonosti

Čas výpočtu (Execution time)

Otázky

- Jak měřit výkon/kvalitu paralelního algoritmu?
- Podle čeho určit nejlepší/nejvhodnější algoritmus pro danou platformu?

Čas výpočtu – sekvenční algoritmus

- Doba, která uplyne od spuštění výpočtu do jeho ukončení.
- T_S

Čas výpočtu – paralelní algoritmus

- Doba, která uplyne od spuštění výpočtu do doby, kdy skončí poslední z paralelních výpočtů.
- Zahrnuje distribuci vstupních i sběr výstupních dat.
- T_P

Celková režie paralelismu (Total overhead)

Celková režie:

- Označujeme T_O
- Veškeré elementy způsobující režii paralelního řešení.
- Celkový čas paralelního výpočtu bez času potřebného pro výpočet problému optimálním sekvenčním řešením.
- Paralelní a sekvenční algoritmy mohou být zcela odlišné.
- Sekvenčních algoritmů může existovat víc.
- T_S čas výpočtu nejlepšího sekvenčního algoritmu (řešení).

Funkce celkové režie

- Jaký je čas výpočtu jednotlivých procesů?
- Doba od skončení výpočtu jednoho procesu do skončení celého paralelního výpočtu se považuje za režii (idling).
- $T_O = pT_P - T_S$

Zrychlení (Speed-up)

Základní přínos paralelizace

- Problémy jdou vždy řešit sekvenčním algoritmem.
- Paralelizací lze dosáhnout pouze zrychlení výpočtu.
- Ostatní výhody jsou diskutabilní a těžko měřitelné.

Základní míra účinnosti samotné paralelizace

- Poměr časů potřebných k vyřešení úlohy na jedné procesní jednotce a p procesních jednotkách.
- Uvažuje se vždy čas nejlepšího sekvenční algoritmu.
- V praxi se často (a nesprávně) používá čas potřebný k vyřešení úlohy paralelním algoritmem spuštěném na jedné procesní jednotce.
- $S = \frac{T_S}{T_P}$.

Zrychlení

Teoretická hranice zrychlení

- S použitím p procesních jednotek je maximální zrychlení p.

Super-lineární zrychlení

- Jev, kdy zrychlení je větší jak p.

Pozorování

- Zrychlení lze měřit i asymptoticky.

Příklad – sčítání n čísel s použitím n procesních jednotek.

- Sekvenčně je potřeba $\Theta(n)$ operací, čas $\Theta(n)$
- Paralelně je potřeba $\Theta(n)$ operací, ale v čase $\Theta(log\ n)$
- Zrychlení $S = \Theta(\frac{n}{log\ n})$

Super-lineární zrychlení

Falešné super-lineární zrychlení

- Uvažme datovou distribuci na 2 procesní jednotky, a operaci kvadratickou ve velikosti dat.
- Zrychlení $S = \frac{n^2}{(\frac{n}{2})^2} = 4$ při použití 2 procesních jednotek.
- Problém – neuvažován optimální sekvenční algoritmus.

Skutečné super-lineární zrychlení

- Větší agregovaná velikost cache pamětí.
- Při snížení množství dat na jeden procesní uzel se účinnost cache pamětí zvýší.
- Výpočet je na $\frac{1}{p}$ datech víc jak p-krát rychlejší.

Super-lineární zrychlení v závislosti na instanci problému

- Průzkumová dekompozice úlohy při hledání řešení.
- Paralelní verze může vykonat menší množství práce.
- V konkrétní instanci lze paralelní prohledávání simulovat i sekvenčním algoritmem, obecně ale nelze.

Amdahlův zákon

Otázka

- Jaké je největší možné zrychlení systému, pokud se paralelizací urychlí pouze část výpočtu?

Amdahlův zákon

- $S_{max} = \frac{1}{(1-P)+\frac{P}{S_P}}$
- P – podíl systému, který je urychlen paralelizací
- S_P – zrychlení dosažené paralelizací nad daným podílem

Příklad

- Paralelizací lze urychlit 4-násobně 30% kódu, tj. $P = 0.3$ a $S_P = 4$.
- Maximální celkové zrychlení S_{max} je

$$S_{max} = \frac{1}{(1-0.3)+\frac{0.3}{4}} = \frac{1}{0.7+0.075} = \frac{1}{0.775} = 1.29$$

Efektivita

Fakta

- Pouze ideální paralelní systém s p procesními jednotkami může dosahovat zrychlení p.
- Část výpočtu prováděná na jednom procesoru je spotřebována režií. Procesor nevěnuje 100% výkonu řešení problému.

Efektivita

- Lze definovat jako podíl zrychlení S vůči počtu jednotek p.
- $E = \frac{S}{p} = \frac{T_s/T_p}{p} = \frac{T_s}{pT_p}$.
- Zrychlení je v praxi $< p$, efektivita v rozmezí $(0, 1\rangle$.
- "*Podíl času, po který jednotka vykonává užitečný kód.*"

Efektivita – příklady

Příklad 1

- Jaká je efektivita sčítání n čísel na n procesorech?
- $S = \Theta(\frac{n}{log\ n})$
- $E = \Theta(\frac{S}{n}) = \Theta(\frac{1}{log\ n})$

Příklad 2

- Na kolik se zkrátí 100 vteřinový výpočet při použití 12 procesorů při 60% efektivitě?
- $E = \frac{T_s}{pT_p}$
- $0.6 = \frac{100}{12x} \implies x = \frac{100}{0.6*12} \implies x = 13.88$

Cena

Cena řešení problému na daném paralelním systému

- Součin počtu procesních jednotek a doby paralelního výpočtu: $C = p \times T_P$
- Označována také jako "množství práce", které paralelní systém vykoná, nebo jako "pT_p produkt".
- Alternativně lze použít pro výpočet účinnosti ($E = \frac{T_S}{C}$).

Cenově optimální paralelní systém

- Cena sekvenčního výpočtu odpovídá nejlepšímu T_S.
- Paralelní systém je cenově optimální, pokud cena řešení roste asymptoticky stejně rychle jako cena sekvenčního výpočtu.
- Cenově optimální systém musí mít efektivitu rovnou $\Theta(1)$.

Příklad

- Sčítání n čísel na n procesorech
- $C = \Theta(n \log n)$, není cenově optimální

Cena – příklad

- Uvažme n procesorový systém, který třídí (řadí) seznam n čísel v čase $(log\ n)^2$
- Nejlepší sekvenční algoritmus má $T_S = nlog\ n$
- $S = \frac{n}{log\ n}$, $E = \frac{1}{log\ n}$
- $C = n(log\ n)^2$
- Není cenově optimální, ale pouze o faktor $log\ n$
- Uvažme, že v praxi $p << n$ (p je mnohem menší než n)
- $T_P = n(log\ n)^2/p$ a tedy $S = \frac{p}{log\ n}$
- Konkrétně: $n = 2^{10}$, $log\ n = 10$ a $p = 32 \Rightarrow S = 3.2$
- Konkrétně: $n = 2^{20}$, $log\ n = 20$ a $p = 32 \Rightarrow S = 1.6$
- Závěr: cenová optimalita je nutná

Vliv granularity na cenovou optimalitu

Vliv granularity na cenovou optimalitu

Tvrzení

- Volbou granularity lze ovlivnit cenu paralelního řešení.

Pozorování

- V praxi $p << n$, přesto navrhujeme algoritmy tak, aby granularita byla až na úrovni jednotlivých položek vstupu, tj. předpokládáme $p = n$.

Deškálování (scale-down)

- Uměle snižujeme granularitu (vytváříme hrubší úlohy).
- Snižujeme overhead spojený s komunikací.
- Může ovlivnit cenu a cenovou optimalitu.

Vliv granularity na cenovou optimalitu

Příklad 1)

- Sčítání n čísel na p procesorech ($n = 16$, $p = 4$)
- Mapování ($i \ mod \ p$), tj. (n/p) čísel na 1 procesor.
- Simulace prvních $log\ p$ kroků stojí $(n/p)log\ p$ operací, tj. proběhne v čase $\Theta((n/p)log\ p)$.
- Následně simulace původního algoritmu probíhá lokálně v paměti jednoho procesoru, tj. v čase $\Theta(n/p)$.
- Celkový T_P je $\Theta((n/p)log\ p)$
- Cena $C = \Theta(nlog\ p)$, tj. není cenově optimální
- Původní T_P bylo $\Theta(log\ n)$, změna o faktor $\frac{n}{p}(\frac{log\ p}{\log\ n})$

Vliv granularity na cenovou optimalitu

Příklad 2)

- Sčítání n čísel na p procesorech ($n = 16$, $p = 4$)
- Nechť n a p jsou mocniny dvojky (n a p jsou soudělné)
- Mapování ($i \ mod \ p$)
- Každá jednotka nejprve sečte data lokálně v čase $\Theta(n/p)$
- Problém redukován na sčítání p čísel na p procesorech
- Celkový T_P je $\Theta(n/p + log\ p)$
- Cena $C = \Theta(n + plog\ p)$,
- Cenově optimální, pokud $n > p\ log\ p$ (cena $C = \Theta(n)$)

Škálovatelnost paralelních programů

Škálovatelnost paralelních programů

Fakta

- Programy jsou testovány na malých vstupních datech na systémech s malým počtem procesních jednotek.
- Použití deškálování zkresluje měření výkonu aplikace.
- Algoritmus, který vykazuje nejlepší výkon na testovaných datech, se může ukázat být tím nejhorším algoritmem, při použití na skutečných datech.
- Odhad výkonu aplikace nad reálnými vstupními daty a větším počtu procesních jednotek je komplikovaný.

Škálovatelnost

- Zachování výkonu a efektivity při zvyšování počtu procesních jednotek a zvětšování objemu vstupních dat.

Závěr

- Je třeba uvážit analytické techniky pro vyhodnocování výkonu a škálování.

Charakteristiky škálování – Efektivita

Efektivita paralelních programů:

$$E = \frac{S}{p} = \frac{T_S}{pT_P} = \frac{1}{1 + \frac{T_O}{T_S}}$$

Celková režie (T_O)

- Přítomnost sekvenční části kódu je nevyhnutelná. Pro její vykonání je třeba čas t_{serial}. Po tuto dobu jsou ostatní procesní jednotky nevyužité.
- $T_O > (p-1)t_{serial}$
- T_O roste minimálně lineárně vzhledem k p, často však i asymptoticky více.

Úloha konstantní velikosti (T_S fixní)

- Se zvyšujícím se p, efektivita nevyhnutelně klesá.
- **Nevyhnutelný úděl všech paralelních programů.**

Charakteristiky škálování – Příklad

Problém

- Úkol: Sečíst n čísel s využitím p procesních jednotek.
- Uvažme cenově optimální verzi algoritmu.
- $T_p = (\frac{n}{p} + log\ p)$.
- Lokální operace stojí 1, komunikace 2 jednotky času.

Charakteristiky řešení

- $T_P = \frac{n}{p} + 2log\ p$
- $S = \frac{n}{\frac{n}{p}+2log\ p}$
- $E = \frac{1}{1+\frac{2plog\ p}{n}}$

Charakteristiky škálování – Škálovatelný systém

Při zachování T_S

- Rostoucí počet procesních jednotek snižuje efektivitu.

Při zachování počtu procesních jednotek

- Rostoucí T_S (velikost vstupních dat) má tendenci zvyšovat efektivitu.

Škálovatelné systémy

- Efektivitu lze zachovat při souběžném zvyšování T_S a p.
- Zajímáme se o poměr, ve kterém se T_S a p musí zvyšovat, aby se zachovala efektivita.
- Čím menší poměr T_S/p tím lepší škálovatelnost.

Izoefektivita jako metrika škálování

Vztahy

- $T_P = \frac{T_S + T_O(W,p)}{p}$
- $S = \frac{T_S}{T_P} = \frac{pT_S}{T_S + T_O(W,p)}$
- $E = \frac{S}{p} = \frac{T_S}{T_S + T_O(W,p)} = \frac{1}{1 + T_O(W,p)/T_S}$

W – velikost vstupních dat

Konstantní efektivita

- Efektivita je konstantní, pouze pokud $T_O(W,p)/T_S$ je konstantní.
- Úpravou vztahu pro efektivitu: $T_S = \frac{E}{1-E} T_O(W,p)$.
- Při zachování míry efektivity lze $E/(1-E)$ označit jako konstantu K.

Funkce izoefektivity (stejné efektivity)

$$T_S = KT_O(W,p)$$

Izoefektivita jako metrika škálování

Funkce izoefektivity

$$T_S = KT_O(W, p)$$

Pozorování

- Při konstantní efektivitě, lze T_S vyjádřit jako funkci p
- Vyjadřuje vztah, jak se musí zvýšit T_S při zvýšení p
- Nižší funkce říká, že systém je snáze škálovatelný
- Isoefektivitu nelze měřit u systémů, které neškálují

Izoefektivita jako metrika škálování – příklad

Příklad

- Součet n čísel na p procesorech
- $E = \frac{1}{1+(2p\,log\,p)/n} = \frac{1}{1+T_O(W,p)/T_S}$
- $T_O = 2p\,log\,p$
- Funkce izoefektivity: $T_S = K2p\,log\,p$
- Funkce izoefektivity: $\Theta(p\,log\,p)$

- Při zvýšení procesních jednotek z p na p' se pro zachování efektivity musí zvětšit velikost problému o faktor $(p'\,log\,p')/(p\,log\,p)$.

Cenová optimalita a izoefektivita

Optimální cena

- Pro cenově optimální systémy požadujeme $pT_P = \Theta(T_S)$.
- Po dosazení ze základních izo vztahů dostáváme:
  $T_S + T_O(W, p) = \Theta(T_S)$

Cenová optimalita může být zachována pouze pokud:

- režie je nejvýše lineární vůči složitosti sekvenčního algoritmu, tj. funkce $T_O(W, p) \in \mathcal{O}(T_S)$
- složitost sekvenčního algoritmu je minimálně tak velká jako režie, tj. funkce $T_S \in \Omega(T_O(W, p))$

Spodní odhad na izoefektivitu

Izoefektivita

- Čím nižší/menší funkce, tím lepší škálovatelnost.
- Snaha o minimální hodnotu izofunkce.

Sublineární izoefektivita

- Pro problém tvořený N jednotkami práce je optimální cena dosažitelná pouze pro maximálně N procesních jednotek.
- Přidáváním dalších jednotek vyústí v idling některých jednotek a snižování efektivity.
- Aby se efektivita nesnižovala musí množství práce růst minimálně lineárně vzhledem k p.
- Funkce izoefektivity nemůže být sublineární.

Minimální doba výpočtu

Otázka

- Jaká je minimální možná doba výpočtu (T_P^{min}) při dostupnosti dostatečného počtu procesních jednotek?

Pozorování

- Při rostoucím p se T_P asymptoticky blíží k T_P^{min}.
- Po dosažení T_P^{min} se T_P zvětšuje spolu s p.

Jak zjistit T_P^{min}?

- Nechť p_0 je kořenem diferenciální rovnice $\frac{dT_P}{dp}$.
- Dosazení p_0 do vztahu pro T_P dává hodnotu T_P^{min}.

Příklad – součet n čísel p procesními jednotkami

- $T_P = \frac{n}{p} + 2log\ p, \quad \frac{dT_P}{dp} = -\frac{n}{p^2} + \frac{2}{p}$
- $p_0 = \frac{n}{2}, \quad T_P^{min} = 2log\ n = \Theta(log\ n)$

Asymptotická analýza paralelních programů

Algoritmus	A1	A2	A3	A4
p	n^2	$log\ n$	n	$\sqrt{n}$
T_P	1	n	$\sqrt{n}$	$\sqrt{n} log\ n$
S	$n\ log\ n$	$log\ n$	$\sqrt{n} log\ n$	$\sqrt{n}$
E	$\frac{log\ n}{n}$	1	$\frac{log\ n}{\sqrt{n}}$	1
pT_P	n^2	$n\ log\ n$	$n^{1.5}$	$n\ log\ n$

Otázky

- Jaký algoritmus je nejlepší?
- Použily jsme vhodné odhady složitosti?
- Je dostupná odpovídající paralelní architektura?

Závěr

Návrh paralelních algoritmů ač složitostně neoptimálních je nedílnou a důležitou částí práce programátora paralelních aplikací.

Shrnutí

Co jsme se v kurzu dozvěděli

Paralelní HW platformy

- Principy fungování HW systémů se sdílenou pamětí.
- Komunikace v systémech s distribuovanou pamětí.

Nástroje paralelního programování

- Standardy a knihovny pro implementaci paralelních aplikací.

  `POSIX Threads, OpenMP, MPI (OpenMPI)`
- Principy fungování těchto knihoven

  `Lock-Free datové struktury, Kolektivní komunikace`

Algoritmizace

- Principy návrhu paralelních algoritmů.
- Analytické posuzování paralelních řešení.