IB109 Návrh a implementace paralelních systémů

Principy návrhu paralelních algoritmů

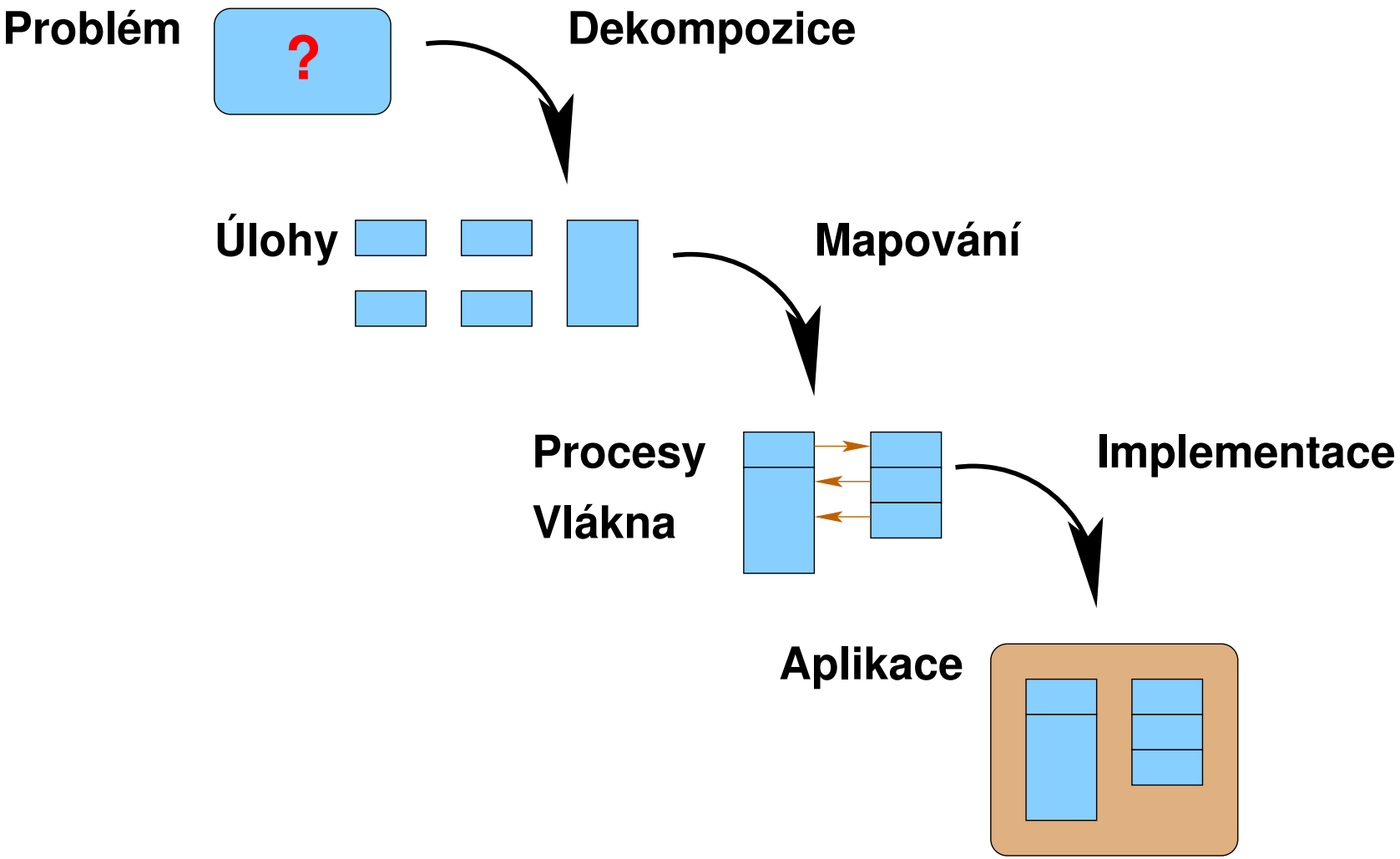
Jiří Barnat

Vícepráce programátora paralelních aplikací

- Identifikovat souběžně proveditelné činnosti a jejich závislosti.
- Mapovat souběžně proveditelné části práce do procesů.
- Zajistit distribuci vstupních, vnitřních a výstupních dat.
- Spravovat souběžný přístup k datům a sdíleným prostředkům.
- Synchronizovat jednotlivé procesy v různých stádiích výpočtu tak, jak vyžaduje paralelní algoritmus.
- Mít znalost přídavných programátorských prostředků související s vývojem paralelních algoritmů.

Základy návrhu paralelních algoritmů

Návrh a realizace paralelního systému



Dekompozice a Úlohy

Dekompozice

- Proces rozdělení celé výpočetní úlohy na podúlohy.
- Některé podúlohy mohou být prováděny paralelně.

(Pod)úlohy

- Jednotky výpočtu získané dekompozicí.
- Po vyčlenění se považují za dále nedělitelné.
- Mají uniformní/neuniformní velikost.
- Jsou definované v době kompilace / za běhu programu.

Příklad

- Násobení matice A $(n \times n)$ vektorem B
- $C[i] = \sum_{j=1}^{n} A[i,j].B[j]$

Závislosti úloh

Graf závislostí

- Zachycuje závislosti prováděných úloh.
- Definuje relativní pořadí provádění úloh (částečné uspořádání).

Vlastnosti a využití grafu

- Orientovaný acyklický graf.
- Graf může být nespojitý, či dokonce prázdný.
- Úloha je připravena ke spuštění, pokud úlohy, na kterých závisí, dokončili svůj výpočet (topologické uspořádání).

Příklady závislostí

- Pořadí oblékání svršků.
- Paralelní vyhodnocování výrazů

  `v AND x AND (y OR z)`

Granularita a Stupeň souběžnosti

Granularita

- Počet úloh, na který se problém dekomponuje.
- Mnoho malých úloh – jemnozrnná granularita (fine-grained).
- Málo větších úloh – hrubozrnná granularita (coarse-grained).
- Každý problém má vnitřní hranici granularity.

Stupeň souběžnosti

- Maximální počet úloh, které mohou být prováděny souběžně.
- Limitem je vnitřní hranice granularity.

Průměrný stupeň souběžnosti

Průměrný stupeň souběžnosti

- Závislý na grafu závislostí a granularitě.
- Mějme množství práce asociované k uzlům grafu.
- **Kritická cesta** – cesta, na které je součet prací maximální.
- **Průměrný stupeň souběžnosti** je podíl celkového množství práce vůči množství práce na kritické cestě.
- Udává maximální zrychlení, pokud je cílová platforma schopna vykonávat souběžně maximální stupeň souběžnosti úloh.

Pozorování

- Zjemňování dekompozice může zvýšit stupeň souběžnosti.
- Čím méně práce je na kritické cestě, tím větší je potenciál pro paralelizaci.

Interakce úloh – Další omezení paralelizace

Interakce úloh

- Nezávislé úlohy mohou vzájemně komunikovat.
- Obousměrná komunikace může snižovat stupeň souběžnosti (úlohy musí co-existovat ve stejný okamžik).
- Komunikace úloh – **neorientovaný graf interakcí**.
- Graf interakcí pokrývá graf závislostí (ověření splnění předpokladů pro spuštění úlohy je forma interakce).

Příklad jednosměrné komunikace

- Násobení matice vektorem ($y = Ab$)
- Dekompozice na nezávislé úlohy dle řádků matice A.
- Prvky vektoru b jsou čteny ze všech úloh, je nutné je vhodně distribuovat k jednotlivým úlohám.

Techniky dekompozice

Techniky dekompozice

Dekompozice

- Fundamentální technika v návrhu paralelních algoritmů.

Obecné dekompozice

- Rekurzivní
- Datová

Specializované dekompozice

- Průzkumová
- Spekulativní
- Hybridní

Rekurzivní dekompozice

Vhodné pro problémy typu rozděl a panuj.

Princip

- Problém se dekomponuje na podúlohy tak, aby jednotlivé úlohy mohly být dekomponovány stejným způsobem jako rodičovská úloha.
- Někdy je třeba restrukturalizovat úlohu.

Příklad

- Quicksort
  - Provede se volba pivota.
  - Rozdělení pole na prvky menší než a větší rovno než.
  - Rekurzivně se opakuje dokud je množina prvků neprázdná.
- Hledání minima v lineárním poli.
  - Princip půlení prohledávaného pole.
  - Typický příklad restrukturalizace výpočtu.

Datová dekompozice

Základní princip

- Data se rozdělí na části (data partitioning).
- Úlohy se provádí souběžně nad jednotlivými částmi dat.

Datová dekompozice podle místa

- Vstupní data
- Výstupní data
- Vnitřní data
- Kombinace

Mapování dat na úlohy

- Funkce identifikující vlákno odpovědné za zpracování dat.

Úlohy typu "Embarrassingly parallel"

- Triviální datová dekompozice na dostatečný počet zcela nezávislých, vzájemně nekomunikujících úloh.

Průzkumová dekompozice

Princip

- Specializovaná technika paralelizace.
- Vhodná pro prohledávací úlohy.
- Prohledávaný prostor se rozdělí podle směru hledání.

Vlastnosti

- Při znalosti prohledávaného stavového prostoru lze dosáhnout optimálního vyvážení a zatížení procesorů.
- Na rozdíl od datové dekompozice, úloha končí jakmile je nalezeno požadované.
- **Množství provedené práce se liší od sekvenční verze.**
- V případě, že graf není strom, je třeba řešit problém opakujících se konfigurací (riziko nekonečného výpočtu).

Příklad

- Řešení hlavolamu "patnáct"

Spekulativní dekompozice

Princip

- Specializovaná technika paralelizace.
- Vhodná pro úlohy se sekvencí datově závislých podúloh.
- Úloha, která čeká na výstup předchozí úlohy, se spustí nad všemi možnými vstupy (výstupy předchozí úlohy).

Vlastnosti

- Provádí se zbytečná práce.
- Nemusí být ve výsledku rychlější jak serializovaná verze.
- Vhodné pro úlohy, kde jistá hodnota mezivýsledku má velkou pravděpodobnost.
- Vzniká potenciální problém při přístupu ke zdrojům (některé zdroje nemusí být sdílené v případě sekvenčního vykonávání úloh).

Příklady

- Spekulativní provádění kódu (větvení).

Hybridní dekompozice

Kombinace různých způsobů dekompozice

Příklad

- Hledání minima v poli.
- Sekvenční verze najde minimum v $O(n)$.
- Při použití datové a rekurzivní dekompozice lze trvání této úlohy zkrátit na $O(n/p + log(p))$.
- Vstupní pole se datově dekomponuje na p stejných částí.
- Najdou se minima v jednotlivých částech v čase $O(n/p)$.
- Výsledky z jednotlivých třídění se zkombinují v čase $O(log(p))$.
- Teoreticky lze při dostatečném počtu procesorů nalézt minimum v čase $O(log(n))$.
- Tento styl paralelismu je obecně označován jako **MAP-REDUCE**.

Techniky mapování a vyrovnávání zátěže

Mapování úloh na vlákna/procesy.

Mapování

- Přiřazování úloh jednotlivým vláknům/procesům.
- Optimální mapování bere v potaz grafy závislostí a interakce.
- Ovlivňuje výkon aplikace.
- Naivní mapování (úloha=proces/vlákno)

Cíle mapování

- Minimalizovat celkový čas řešení celé úlohy.
  - Redukovat prodlevy způsobené čekáním (idling)
  - Redukovat zátěž způsobenou interakcí
  - Redukovat režii spouštění, ukončování a přepínání
  - Vyrovnat zátěž na jednotlivé procesory
- Maximalizovat souběžnost.
- Minimalizovat zatížení systému (zatížení datových cest).
- Využít dostupnost zdrojů použitých předchozí úlohou.

Charakteristiky úloh, které ovlivňují mapování

Způsob zadání úlohy

- **Statické zadání úloh** – dekompozice problému na úlohy je dána v době kompilace, případně je přímo odvozena od vstupních dat.
- **Dynamické zadání úloh** – nové úlohy jsou vytvářeny za běhu aplikace dle průběhu výpočtu, případně jako důsledek provádění původně zadaných úloh.

Velikost úlohy

- Relativní množství času potřebné k dokončení úlohy.
- Uniformní vs. neuniformní.
- Dopředná znalost/neznalost.

Velikost dat asociovaných k úloze

- Snaha o zachování lokality dat.
- Různá data mají různou roli a velikost (vstupní/výstupní data u hlavolamu patnáct).

Charakteristiky interakcí, které ovlivňují mapování

Statické vs Dynamické

- Statické: Probíhají v předdefinovaném časovém intervalu, mezi předem známou množinou úloh.
- Dynamické: Pokud předem neznáme počet interakcí, časový rámec interakcí, nebo participující úlohy.

Další charakteristiky

- Jednosměrná versus obousměrná interakce.
- Mód přístupu k datům: Read-Only versus Read-Write.
- Pravidelné versus nahodilé interakce.

Režie související s mapováním do různých vláken/procesů

- Uzpůsobení aplikace pro neočekávanou interakci.
- Připravenost dat k odeslání / adresáta k přijetí.
- Řízení přístupu ke sdíleným zdrojům.
- Optimalizace aplikace pro redukci prodlev.

Schémata pro Statické Mapování

Mapování založené na rozdělení dat

- Bloková distribuce
- Cyklická a blokově-cyklická distribuce
- Náhodná distribuce bloků
- Dělení grafu

Mapování založené na rozdělení úloh

- Dělení dle grafu závislostí úloh
- Hierarchické dělení

Mapování založené na rozdělení dat

Bloková distribuce datových polí

- Procesy svázány s daty rozdělenými na souvislé bloky.
- Bloky mohou být vícerozměrné (redukce interakcí).
- Příklad
  - Násobení matic $A \times B = C$
  - Dělení matice C na 1- a 2-rozměrné bloky.

Cyklická a blokově-cyklická distribuce datových polí

- Nerovnoměrné množství práce spojené s jednotlivými prvky
- $\Rightarrow$ blokové dělení způsobuje nerovnoměrné zatížení.
- Blokově-cyklická distribuce: dělení na menší díly a cyklické přiřazení procesům (round robin).
- Zmenšování bloků vede k cyklické distribuci (blok je atomický prvek datového pole).

Mapování založené na rozdělení dat – pokračování

Náhodná distribuce bloků

- Zátěž související s prvky pole vytváří pravidelné vzory.
- $\Rightarrow$ špatná distribuce v cyklickém rozdělení.
- Náhodné přiřazení bloků procesům.

Grafové dělení

- Pro případy, kdy je nevhodné organizovat data do polí (například drátové modely 3D objektů).
- Data organizována jako graf.
- Optimální dělení.
  - Stejný počet vrcholů v jednotlivých částech.
  - Co možná nejmenší počet hran mezi jednotlivými částmi.
  - NP-úplný problém.

Mapování založené na rozdělení úloh

Princip

- Graf závislostí úloh.
- Grafové dělení (NP-úplné).

Speciální případy pro konkrétní tvar grafů

- Binární strom (rekurzivní dekompozice).

Hierarchické mapování

- Úlohové dělení nebere v potaz neuniformitu úloh.
- Shlukování úloh do nad-úloh.
- Definuje hierarchie (vrstvy).
- Jiné mapovací a dekompoziční techniky na jednotlivých vrstvách.

Schémata pro Dynamické Mapování

Motivace

- Statické mapování nedostatečné, neboť charakteristiky úloh nejsou známy v době překladu.

Centralizovaná schémata dynamického mapování

- Úlohy jsou shromažďovány v jednom místě.
- Dedikovaná úloha pro přiřazování úloh procesům.
- Samo-plánování
  - Jakmile proces dokončí úlohu, vezme si další.
- Blokové plánování
  - Přístup ke shromaždišti úloh může být úzkým místem,
  - $\Rightarrow$ přidělování úloh po blocích.

Příklad

- Třídění prvků v $n \times n$ matici A
- `for (i=1; i<n; i++) newtask(sort(A[i],n));`

Schémata pro Dynamické Mapování – pokračování

Distribuovaná schémata

- Množina úloh je distribuována mezi procesy.
- Za běhu dochází k vzájemnému vyměňování úloh.
- Netrpí nedostatky spojenými s centralizovaným řešením.

Možnosti

- Jak se určí, kdo komu pošle úlohu.
- Kdo a na základě čeho určí, že je potřeba přesunout úlohu.
- Kolik úloh má být přesunuto.
- Kdy a jak je úloha přesunuta.

Problém

- Efektivita přenosu úlohy na jiný proces.

Afinitní plánování

Vlákna a procesory

- Jednotlivá vlákna jsou vykonávány fyzickými procesory.
- Plánování zajišťuje plánovač OS.

Afinitní plánování (angl. affinity scheduling)

- Modifikace algoritmu plánování.
- Afinitní plánování zajišťuje, že výpočetní dávky přidělené jednomu procesu/vláknu budou pokud možno přiděleny na fyzicky tentýž procesor.

Výhody a rizika

- Potencionálně lepší využití cache.
- Striktní lpění na tomtéž procesoru může narušovat vyváženost využití procesorů, tedy redukovat výkon aplikace.

Dilema procesu dekompozice-mapování

Otázka

- Je lepší nejprve dekomponovat na mnoho malých úloh a pak úlohy shlukovat při mapování, nebo naopak omezit dekompozici, aby mapování bylo přímočaré?

Aspekty napomáhající rozhodnutí dilematu

- Je cena dekompozice shodná pro oba scénáře?
- Vytváří jemnější dekompozice skutečně nezávislé úlohy?
- Je jemnější dekompozicí zachována datová lokalita?
- Je/není znám počet jader na cílové platformě?
- Jaká je cena režie přepínání, zejména v situaci, kdy počet vláken výrazně převyšuje počet výpočetních jader?

Metody pro redukci režie interakce

Metody pro redukci režie interakce

Režie související s interakcí

- Režie související s interakcí souběžných úloh je klíčovým faktorem ovlivňující výkon paralelní aplikace.
- Z pohledu režie interakce jsou ideální "Embarrassingly parallel" úlohy, kde k interakci nedochází.

Faktory ovlivňující režii

- Objem přenášených dat
- Frekvence interakce
- Cena komunikace

Zvyšování datové lokality

Cíl – snížit objem přenášených dat

- Přesun sdílených datových struktur do lokálních kopií.
- Minimalizace objemu sdílených dat.
- Lokalizace výpočtu (lokální ukládání mezivýsledků).
- Režie protokolů pro udržení koherence lokálních kopií.

Cíl – snížit frekvenci interakce

- Prostorová lokalizace přenášených dat
- Přenáčení dat a jejich okolí (princip cache)
- Více zpráv v jedné (bufferování)

Minimalizace současných přístupů

Problém – Contention

- Přístup k omezenému zdroji ve stejný okamžik (contention) je řešen serializací požadavků.
- Serializace požadavků způsobuje prodlevy.

Možné řešení

- Je potřeba N souběžných přístupů k datům.
- Přistupovaná data je možné rozdělit do N bloků.
- A data číst v N po sobě jdoucích iteracích.
- V každé iteraci je každý blok čten jiným vláknem.
- Číslo čteného bloku v r-té iteraci j-tým vláknem:
  $(r + j) modulo\ N$

Překrývání výpočtu s interakcí

Problém

- Čekání na příjem či odeslání dat způsobuje nechtěné prodlevy.

Včasné vykonání akce – podmínky proveditelnosti

- Data musí být včas připravena.
- Přijímací i odesílací strany mohou asynchronně komunikovat.
- Existuje další úloha, která může být řešena po dobu komunikace.

Jiná řešení

- Simulace mechanismu přerušení (ala operační systém).
- Žádné, kvůli režii způsobené násilným řešením.

Replikace dat či výpočtů

Problém

- Opakované drahé přístupy ke sdíleným datům.

Řešení pro read-only data

- Při prvotní interakci tvorba kopii dat (datová lokalita).
- Dále pracovat s lokální kopií.
- Zvyšuje paměťové nároky výpočtu.

Řešení pro read-write data

- Podobně jako v read-only případě.
- Násobné souběžné výpočty téhož mohou být rychlejší, než čtení a zápis sdílené hodnoty.

Optimalizované operace pro kolektivní komunikaci

Problém

- Stejná interakce mezi všemi procesy vykonávaná základními komunikačními primitivy je drahá.

Řešení – kolektivní komunikační operace

- Pro přístup k datům jiných vláken/procesů.
- Důležité pro komunikačně intenzivní výpočty.
- Forma efektivní synchronizace.

Optimalizované implementace

- MPI

Překrývání Interakcí

Problém

- Nedostatečná propustnost komunikační sítě, či absence kolektivních komunikačních operací.

Řešení

- Zvýšit využití komunikační sítě současnou komunikací mezi různými páry procesů.

Příklad

- 4 procesy $P_1, \ldots, P_4$
- P_1 chce všem poslat zprávu m_1
- $P_1 \rightarrow P_2$, $P_2 \rightarrow P_3$, $P_3 \rightarrow P_4$
- $P_1 \rightarrow P_2$, $P_2 \rightarrow P_3$

  $P_1 \rightarrow P_4$

Komunikace v nesdíleném adresovém prostoru

- Synchronizace posíláním zpráv.
- Předávání dat posíláním zpráv.

Komunikace ve sdíleném adresovém prostoru

- Synchronizace korektním přístupem ke sdíleným datům.
- Předávání dat pomocí sdílených datových struktur. (FIFO)

Obecné charakteristiky

- Latence – doba potřebná pro doručení prvního bitu.
- Přenosová rychlost – objem dat přenesených za jednotku času.

Latence

- Celková cena pro zahájení komunikace — t_s
  - čas pro přípravu zprávy/dat
  - identifikace adresáta / routování
  - doba trvání vylití informace z cache do paměti případně na síťové rozhraní
- Cena "hopů" (přeposílání uvnitř komunikační sítě) — t_h
  - čas strávený na jednotlivých routerech v síti
  - doba, po kterou putuje hlavička zprávy z přijímacího na odesílací port

Přenosová rychlost

- Ovlivněno šířkou pásma r
- Cena za přenos jednoho slova (word = 2 bajty) — $t_w = 1/r$

Cena komunikace

- m – délka zprávy ve slovech
- l – počet linek, přes které zpráva putuje
- $t_s + l * (t_h + mt_w)$

Obecné metody redukce ceny

- Spojování malých zpráv (amortizuje se hodnota t_s)
- Komprese (snižování hodnoty m)
- Minimalizace vzdálenosti (snižování hodnoty l)
- Paketování (eliminace režie způsobené jednotlivými hopy)

$$t_s + l * (t_h + mt_w) \quad \longrightarrow \quad t_s + lt_h + mt_w$$