

Návrh paralelního systému, úvod do distribuovaných systémů

Paralelní a distribuované systémy, přednáška 5

Tomáš Urbanec

Katedra informatiky PřF UPOL

21.10.2025

Co nás čeká?

1. Dokončení části k paralelním systémům
 - Návrh a časté vzory paralelních systémů.
 - Fosterova metodologie.
 - Příklad.
2. Úvod do distribuovaných systémů
 - Co je to distribuovaný systém?
 - Architektury DS.

Návrh paralelního systému

Dekompozice

- = První krok návrhu paralelního systému.
(Reálně až druhý → první vymyslet sekvenční řešení.)
- (Zatím) neřešíme technikálie, jen koncepci.
- 1. Dle úkolů (*task decomposition*):
 - Jaké úkoly (*task*) máme v našem sekvenčním řešení?
 - Najít mezi úkoly závislosti.
 - Graf závislosti úkolů (*task dependency graph*) apod.
 - Náhled na náš proces/program.
 - Limity na konkurentní části programu.

Dekompozice

- = První krok návrhu paralelního systému.
(Reálně až druhý → první vymyslet, jak to vyřešit sekvenčně.)
 - (Zatím) neřešíme technické, jen koncepci.
- 2. Dle dat (*data decomposition*):
 - Jak *data* můžeme rozdělit pro konkurentní zpracování?
 - Potřebuje podporu na nižší vrstvě (HW, DS)
 - Typické pro SIMD.
 - Dobrá horizontální škálovatelnost.

Pipeline

Vzory dekompozice

- Vzor pro dekompozici podle úkolů.
- Úkol se dá rozdělit na více po sobě následujících podúkolů.
- Data jsou zpracovávána postupně.
- Více úkolů běží najednou na jiných částech dat.
- Příklad: montážní linka v továrně na auta, Extract-Transform-Load (ETL), navazující operace na streamu dat, ...
- Procesy (jednotlivé kroky) musíme propojit (fronty, kanály, ...)
- Rozdíl paralelní vs pipeline? Zdroje (počet vláken/procesů).

(tabule)

Paralelizace cyklů (loop-level)

Vzory dekompozice

- Vzor pro dekompozici podle dat.
 - Cyklus opakující stejný úkol pro iterovaná data (nezávisle).
- (Všechny) Iterace lze provést paralelně (nezávislost).
- Obvykle snadná implementace (třeba s ThreadPolem).
 - Kompilátor může provádět automaticky.

Mapování

Vzory dekompozice

- Vzor pro dekompozici podle dat.
 - Máme kolekci dat, kde na každém prvku provádíme stejnou operaci (nezávisle).
- Výpočet s každým prvkem lze provést paralelně (nezávislost).
- Tj. jako paralelizace cyklů ale z funkcionálního pohledu.
 - Často deklarativní přístup.
 - Obojí stojí na nezávislosti iterací/operací na prvcích.

Fork/Join

Vzory dekompozice

- Vzor pro dekompozici podle dat.
- Část zpracování dat se dá udělat paralelně na částech dat.
- Poté potřebujeme zpracovat výsledky z těchto částí.
- Fork - vytvoříme vlákna/procesy pro paralelní zpracování částí.
- Join - počkáme na všechny výsledky a pak s nimi pracujeme dále.
- Příklad: Paralelní merge sort

Map/Reduce

Vzory dekompozice

- Vzor pro dekompozici podle dat.
- První mapování (tj. předchozí slide).
- Poté agregace (reduce).
- Oboje lze provádět paralelně na částech vstupu.
- Agregace výsledků jednotlivých reduce do finálního výsledku (např. další reduce).
- Podobné myšlenky Fork/Join ale funkcionální pohled.
- Často deklarativní přístup:
 - Řekneme co mapovat a čím redukovat.
 - Detaily vyřeší platforma/systém.
- PŘ. Extrémy ve velké kolekci, četnosti něčeho ve velké kolekci,
...

Dekompozice

Další poznámky

- Často v systému máme obojí:
 - dekompozici dle úkolů,
 - dekompozici dle dat.
 - S dekompozicí souvisí i (efektivní) využití zdrojů.
 - málo úkolů a hodně zdrojů?
 - hodně úkolů a málo zdrojů?
 - Příliš hrubá dekompozice nás může později omezit.
- Dekompozici udělat co nejjemnější.
- Při plánování běhu pak lze upravit dle potřeby.

Fosterova metodologie

- Ian Foster, 1995
- Postup návrhu paralelního programu
 1. Dekompozice - rozdělení úkolu na menší části
 2. Komunikace - popis a zabezpečení komunikace mezi částmi
 3. Aglomerace - shlukování částí do logických skupin
 4. Mapování na zdroje - technická realizace
- Nejprve řešíme problém nezávisle na technické realizaci (1 - 2).
- Až pak technické detaily (3 - 4).

(tabule)

Násobení matic

Příklady návrhu PS

- $C = A \cdot B$ (tabule)

Násobení matic

Příklady návrhu PS

- $C = A \cdot B$ (tabule)
0. Sekvenční algoritmus.

Násobení matic

Příklady návrhu PS

- $C = A \cdot B$ (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Nejjemnější rozdělení?

Násobení matic

Příklady návrhu PS

- $C = A \cdot B$ (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Nejjemnější rozdělení?
 - Jednotlivé hodnoty v C .

Násobení matic

Příklady návrhu PS

- $C = A \cdot B$ (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Nejjemnější rozdělení?
 - Jednotlivé hodnoty v C .
 2. Komunikace
 - Co musíme mít k dispozici?
 - Celou matici?
 - Musíme kopírovat? Stačí sdílet?

Násobení matic

Příklady návrhu PS

- $C = A \cdot B$ (tabule)
- 0. Sekvenční algoritmus.
- 1. Dekompozice
 - Nejjemnější rozdělení?
 - Jednotlivé hodnoty v C .
- 2. Komunikace
 - Co musíme mít k dispozici?
 - Celou matici?
 - Musíme kopírovat? Stačí sdílet?
- 3. Aglomerace
 - Zvážit dosud získané.
 - Jakou máme architekturu?
 - Rozumné množství rozumně komunikujících částí.
 - Cíl je efektivita. Ale nepřehnat to - nové zdroje?

Násobení matic

Příklady návrhu PS

- $C = A \cdot B$ (tabule)
- 0. Sekvenční algoritmus.
- 1. Dekompozice
 - Nejjemnější rozdělení?
 - Jednotlivé hodnoty v C .
- 2. Komunikace
 - Co musíme mít k dispozici?
 - Celou matici?
 - Musíme kopírovat? Stačí sdílet?
- 3. Aglomerace
 - Zvážit dosud získané.
 - Jakou máme architekturu?
 - Rozumné množství rozumně komunikujících částí.
 - Cíl je efektivita. Ale nepřehnat to - nové zdroje?
- 4. Mapování
 - Výsledek předchozího kroku přiřadíme na dostupný HW.
 - Může být přenecháno plánovači nebo OS.

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)
0. Sekvenční algoritmus.

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Dle úkolů: rozdělit na slova, spočítat výskyty slova.
 - Dle dat: lze udělat pro části nezávisle a pak zkombinovat.
 - Vhodná úloha pro Map/Reduce přístup.
 - Jednotlivé soubory, odstavce, řádky, ... rozdělím (map) a zagerguji (reduce) a výsledky znovu agreguji (reduce).

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)

0. Sekvenční algoritmus.

1. Dekompozice

- Dle úkolů: rozdělit na slova, spočítat výskyty slova.
 - Dle dat: lze udělat pro části nezávisle a pak zkombinovat.
- Vhodná úloha pro Map/Reduce přístup.
- Jednotlivé soubory, odstavce, řádky, ... rozdělím (map) a zagerguji (reduce) a výsledky znovu agreguji (reduce).

2. Komunikace

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Dle úkolů: rozdělit na slova, spočítat výskyty slova.
 - Dle dat: lze udělat pro části nezávisle a pak zkombinovat.
 - Vhodná úloha pro Map/Reduce přístup.
 - Jednotlivé soubory, odstavce, řádky, ... rozdělím (map) a zagerguji (reduce) a výsledky znovu agreguji (reduce).
 2. Komunikace
 - Rozdělení a načtení úvodních (velkých) dat.
 - Předání mezivýsledků do reduce.
 - Předání mezivýsledku do dalšího reduce.

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Dle úkolů: rozdělit na slova, spočítat výskyty slova.
 - Dle dat: lze udělat pro části nezávisle a pak zkombinovat.
 - Vhodná úloha pro Map/Reduce přístup.
 - Jednotlivé soubory, odstavce, řádky, ... rozdělím (map) a zagerguji (reduce) a výsledky znovu agreguji (reduce).
 2. Komunikace
 - Rozdělení a načtení úvodních (velkých) dat.
 - Předání mezivýsledků do reduce.
 - Předání mezivýsledku do dalšího reduce.
 3. Aglomerace

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Dle úkolů: rozdělit na slova, spočítat výskyty slova.
 - Dle dat: lze udělat pro části nezávisle a pak zkombinovat.
 - Vhodná úloha pro Map/Reduce přístup.
 - Jednotlivé soubory, odstavce, řádky, ... rozdělím (map) a zagerguji (reduce) a výsledky znovu agreguji (reduce).
 2. Komunikace
 - Rozdělení a načtení úvodních (velkých) dat.
 - Předání mezivýsledků do reduce.
 - Předání mezivýsledku do dalšího reduce.
 3. Aglomerace
 - Map a první reduce lze udělat najednou (ušetřím komunikaci)

Četnost slov v textu

Příklady návrhu PS

- Pro každé slovo v textu určit počet jeho výskytů. (tabule)
0. Sekvenční algoritmus.
 1. Dekompozice
 - Dle úkolů: rozdělit na slova, spočítat výskyty slova.
 - Dle dat: lze udělat pro části nezávisle a pak zkombinovat.
 - Vhodná úloha pro Map/Reduce přístup.
 - Jednotlivé soubory, odstavce, řádky, ... rozdělím (map) a zagerguji (reduce) a výsledky znovu agreguji (reduce).
 2. Komunikace
 - Rozdělení a načtení úvodních (velkých) dat.
 - Předání mezivýsledků do reduce.
 - Předání mezivýsledku do dalšího reduce.
 3. Aglomerace
 - Map a první reduce lze udělat najednou (ušetřím komunikaci)
 4. Mapování
 - Plánovač.
 - Systémy typu MapReduce, Spark, ...?

Distribuované systémy

Paralelní vs distribuovaný systém

- Distribuovaný systém lze chápat jako speciální případ paralelního systému.
- Více (autonomních) systémů spolupracujících na jednom úkolu.
- Z venku se jeví jako jeden systém.
- Mohou být na jednom uzlu/stroji, na LAN i WAN.
- Může to být více distribuovaných systémů tvářících se jako jeden (větší) systém.
- Rozdíly oproti dosud uvažovaným paralelním systémům:
 - Absence sdílené paměti.
 - Komunikace (po síti? spolehlivost? rychlost? latence?).
 - Absence synchronizace času.
 - Problémy jsou obvykle složitější.
 - ...

Příklady

Distribuované systémy

- Aplikace komunikující s databází (frontend, backend, DB, ...).
- Portál, STAG, ...
- Steam, WoW, LoL, CS, Valorant, ...
- Web
- IoT
- Blockchain
- Internet

Architektura distribuovaných systémů

Architektura DS

- software vs. systémová architektura
 - Jak vypadá struktura a rozhraní jednotlivých komponent?
 - Jsou provázané?
 - Jak budou komponenty rozloženy do sítě?
- Mnoho různých možností (dále)
- Často se v nich prolínají oba pohledy.

Klient/Server

Architektura DS

- Klienti (uživatelé, aplikace, ...) požadují službu od serverů. Servery služby nabízejí.
- Jeden uzel může být pro někoho server a pro někoho klient.
- Typicky (ale ne nutně) po síti.
- Centralizované okolo serveru.
- Klienti typicky jednoduché procesy, řeší zejména interakci se serverem (API).
- Request-response (požadavek-odpověď) komunikace. Často asynchronní.
- Př. webové služby, souborové servery, databázové servery, emailové servery, ...

Peer-to-Peer (P2P)

Architektura DS

- ≈ Extrémní verze myšlenky „uzel může být klient i server“
- Každý uzel má stejnou roli i odpovědnost.
 - Každý uzel zná své sousedy.
 - Plně decentralizované.
 - Snadno škálovatelné.
 - Velmi dynamické (uzly mizí a přibývají dle libosti).
 - PŘ. sdílení souborů, Blockchain, BitTorrent (?)

Vrstvená architektura

Architektura DS

= Layered

- Systém je rozdělený do logických vrstev.
- Velmi často 3 vrstvy:
 - Uživatelské rozhraní
 - Logika aplikace
 - Datová vrstva
- Vrstvy odděleny (nezávislé) a komunikují mezi sebou (API).
- Komunikace typicky jen se sousedními vrstvami (ne vždy).
- Modulární, *teoreticky* dobře udržitelné.
- Př. typická větší webová aplikace (FE v JS, BE v ?, data v databázi/ích)

Architektura zaměřená na služby

Architektura DS

= Service oriented architecture

- Snaha o znovupoužitelnost komponent jako služeb.
- Služby mohou běžet na různých strojích, vystavují rozhraní (API).
- Vysoká modularita.
- Dobrá škálovatelnost.

Mikroslužby

Architektura DS

= Microservice

≈ Neprovázaná SOA

- Různé microservice by mělo být možné nasazovat nezávisle.
- Obecná SOA může vyžadovat nasazení/úpravu více služeb najednou (provázanost).
- Členění spíše podle bussiness logiky než podle klasických vrstev či technologií.

Architektura zaměřené na události

Architektura DS

- = Event based architecture / Event driven architecture / Publish-subscribe
 - Centrální prvek jsou události a jejich průběh systémem.
 - Komponenty asynchronně produkují a konzumují události.
 - Komponenty na sobě plně nezávislé.
 - Postaveno okolo systému, který události sbírá a distribuuje.
 - Př. chytrá domácnost, obecně IoT, Kafka...

Hybridní architektura a další

Architektura DS

- Hybridní = kombinace více přístupů
- Příklad: BitTorrent - P2P s prominentními uzly majícími roli serverů
- Příklad: Větší systém, používající microservice pro pomocné služby, client-server pro komunikaci s uživatelem a P2P pro sdílení obsahu mezi uživateli.
- Další:
 - Message oriented,
 - Leader-Follower (Main-replica),
 - Middleware,
 - ...

Changelog