

Klasické synchronizační problémy, návrh paralelního systému

Paralelní a distribuované systémy, přednáška 4

Tomáš Urbanec

Katedra informatiky PřF UPOL

14.10.2026

Co nás čeká?

- Tři základní synchronizační problémy:
 1. Producent a konzument
 2. Čtenáři a písaři
 3. Večeřící filozofové
- Návrh paralelního systému
- Na cvičení se podíváme ještě na
 - problém kuřáků cigaret
 - pár příkladů návrhu PS.
- Většina přednášky proběhne na tabuli.
- Přednáška $\pm$ následuje kapitolu 4 z [1]. Doporučuji pročíst.

Producent a konzument

Producent a konzument

- Často se vyskytující problém.
- Dva typy procesů:
 - Producent - produkuje data, události, úkoly, ...
 - Konzument - konzumuje (a dále zpracovává) data, události, úkoly, ...
- Producentů a/nebo konzumentů je typicky více.
- Komunikují přes buffer.
- Každá položka je vyprodukována a zkonsumována právě jednou.
- Cílem bufferu **není** řešit rozdíl v rychlosti procesů.
- Příklady: event loop, stream, obecně zpracovávání výsledků průběžně generujících procesů

Producent a konzument

Podmínky řešení

- Když někdo přidává/vyzvedává data do/z bufferu, buffer není v konzistentním stavu.
- Přístup k bufferu musí být ve vzájemném vyloučení.
- Pokud konzument chce data a buffer je prázdný, tak konzument musí počkat.
- (Rozbor řešení na tabuli.)

Producent a konzument s konečným bufferem

Podmínky řešení

- Reálně buffer není nekonečný - má kapacitu n prvků.
- To mění situaci.

Upravené podmínky řešení:

- Když někdo přidává/vyzvedává data do/z bufferu, buffer není v konzistentním stavu.
- Přístup k bufferu musí být ve vzájemném vyloučení.
- Pokud konzument chce data a buffer je prázdný, tak konzument musí počkat.
 - Pokud producent chce přidat data a buffer je plný, tak producent musí počkat.

(Rozbor řešení na tabuli.)

Producent a konzument
oooo

Čtenáři a písáři
●oooo

Večeřící filozofové
ooo

Další problémy
oo

Návrh paralelního systému
oooooooooooooooo

Čtenáři a písáři

Čtenáři a písáři

- Často se vyskytující problém.
- Dva typy procesů:
 - Písáři - písáři zapisují do dat.
 - Čtenáři - čtenáři čtou z dat
- Písáři musí psát ve vzájemném vyloučení.
- Čtenáři mohou číst zároveň.
- Každý písář je ve vzájemném vyloučení se všemi čtenáři.
- Příklady: přístup k souboru, DB, datové struktury, ...

Čtenáři a písaři

Podmínky řešení

- Libovolné množství čtenářů může být v kritické sekci (tj. pracovat se zdrojem) zároveň.
- Každý písař musí být v kritické sekci sám (tj. mít při práci výhradní přístup k datům).

(Rozbor řešení na tabuli.)

Féroví čtenáři a písaři

Podmínky řešení

- Písař může vyhladovět, pokud vždy čte nějaký čtenář (střídají se).
- Chceme, aby písaři měli přednost. Tj. čeká-li písař, noví čtenáři nemohou začít číst.
- To mění situaci.

Féroví čtenáři a písáři

Upravené podmínky řešení

- Libovolné množství čtenářů může být v kritické sekci (tj. pracovat se zdrojem) zároveň.
- Každý písář musí být v kritické sekci sám (tj. mít při práci výhradní přístup k datům).
- Pokud chce zapisovat nějaký písář, tak nesmí začít číst nový čtenář. Čtenáři, kteří již čtou, mohou v klidu dočíst.

(Rozbor řešení na tabuli.)

Čtenáři a písáři

Poznámky

- Rozumný požadavek je i prioritou písářů. Tj. čeká-li písář i čtenář, písář má vždy přednost.
- Řešení není složité, promyslete a pak vezměte knížku.

Rozdíly mezi problémy producenti-konzumenti a čtenáři-písáři:

Producenti-konzumenti	Čtenáři-písáři
sdílený buffer	sdílený zdroj (DB, soubor, ...)
pře/podtečení bufferu	konzistence při čtení/zápisu
vzájemné vyloučení všech	vzájemné vyloučení mezi třídami
vyhladovění nehrozí*	musíme si dávat pozor

* předpokládáme alespoň slabou férovost

Producent a konzument
oooo

Čtenáři a písáři
oooooo

Večeřící filozofové
●oo

Další problémy
oo

Návrh paralelního systému
oooooooooooooooo

Večeřící filozofové

Večeřící filozofové

- Často se vyskytující problém.
- Procesy požadující více sdílených zdrojů (konkurují si).
- Jeden typ procesu - filozof - se dvěma operacemi:
 - jí (vyžaduje zdroje),
 - přemýšlí (nevyžaduje zdroje)
- Filozofové sedí u kulatého stolu a mezi každou dvojicí leží jedna vidlička.
- Každý filozof k jídlu potřebuje obě přilehlé vidličky (levou a pravou).

Večeřící filozofové

Podmínky řešení

- Každou vidličku může v jednom okamžiku držet nejvýše jeden filozof.
- Vždy někdo může jíst (nesmí dojít k deadlocku).
- Nikdo nesmí umřít hlady (proces nesmí vyhladovět).
- Musíme umožnit, aby v jednom okamžiku jedlo více filozofů (musíme být efektivní).

(Rozbor řešení na tabuli.)

Producent a konzument
oooo

Čtenáři a písáři
oooooo

Večeřící filozofové
ooo

Další problémy
●o

Návrh paralelního systému
oooooooooooooooo

Další problémy

Další (ne)klasické problémy

- Modelují typické problémy, na které v nějaké formě narazíte v praxi a poskytují standardní a prověřené řešení.
- Dobrá zábava na dlouhé zimní večery.

Několik příkladů:

- kuřáci cigaret (cvičení),
- spící holič,
- hodující divoši,
- H_2O ,
- Santa Claus,
- ...

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 technikálie, 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, ...?



A. B. Downey. 2016. *The Little Book of Semaphores*. Druhé vydání. Dostupné z <http://greenteapress.com/semaphores>.

Producent a konzument
oooo

Čtenáři a písáři
oooooo

Večeřící filozofové
ooo

Další problémy
oo

Návrh paralelního systému
oooooooooooo●

Changelog