

PB165 Grafy a sítě: Plánování na počítačové síti II.

- 1 Popis problému
- 2 Plánování seznamem
- 3 Heuristiky mapování
- 4 Shlukovací heuristiky

Minulá přednáška:

Plánování komunikace současně běžících úloh: paralelně komunikující úlohy

- plánování v daném okamžiku
- plánování strojů a komunikace pro úlohy

Plánování současně probíhajících přenosů

- plánování všech komunikací pro aktuální okamžik
- plánování strojů neuvažováno
- uvažována topologie počítačové sítě

Dnešní přednáška:

Plánování úloh s precedencemi i bez precedencí

- uvažována topologie počítačové sítě
- plánování strojů pro úlohy
- plánování přenosu dat po ukončení úlohy jiné úloze (úlohám)

Příklady aplikací

- plánování výpočetních úloh na počítače
 - plánování úloh vyžadujících procesory, disky, sw. licence, ...
 - precedenční závislosti mezi úlohami méně časté
- plánování acyklických workflows
 - acyklický graf precedenčních závislostí mezi úlohami
 - přenos dat po skončení úlohy následující úlohám

Algoritmy

- optimální (polynomiální složitost)
 - použitelné pouze pro velmi specializované problémy
 - obtížná rozšiřitelnost na obecné problémy
 - př. plánování stromů, plánování na dvou procesorech
- heuristické
 - plánování seznamem (list scheduling)
 - heuristiky mapování (mapping heuristics)
 - shlukovací heuristiky (clustering heuristics)

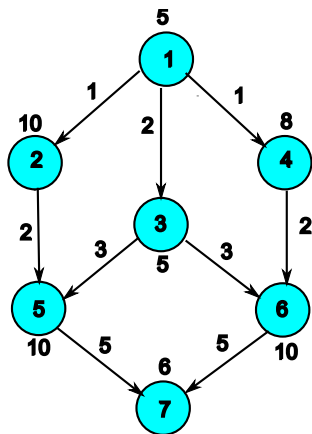
Úlohy a komunikace

Referenční doba trvání p_j úlohy j

Precedenční omezení $j_1 \rightarrow j_2$

Komunikace: $size_{j_1,j_2}$

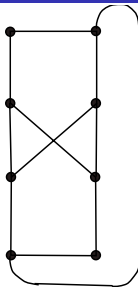
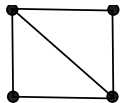
- $size_{j_1,j_2}$: velikost přenesených dat mezi úlohami j_1 a j_2
- pokud existuje $j_1 \rightarrow j_2$, pak $size_{j_1,j_2} \geq 0$, jinak $size_{j_1,j_2} = 0$
- pokud jsou j_1 a j_2 zpracovány na stejném stroji, tak lze čas na komunikaci (a tedy i velikost přenášených dat) zanedbat



Hranově a vrcholově ohodnocený acyklický orientovaný graf

- vrcholy: úlohy
- orientované hrany: precedenční vztahy mezi úlohami
- ohodnocení hrany: velikost dat na komunikaci
- ohodnocení vrcholu: referenční doba trvání

- Topologie sítě: příklady



každý uzel grafu reprezentuje dual-core procesor

- $transrate_{i1,i2}$: přenosová rychlost mezi sousedními stroji $i1, i2$
- $setupmesg_i$: (inicializační) doba na poslání zprávy strojem i
- Komunikační zpoždění $c(j1, j2, i1, i2)$ pro poslání dat z úlohy $j1$ úloze $j2$ ze stroje $i1$ na sousední stroj $i2$

$$c(j1, j2, i1, i2) = \frac{size_{j1,j2}}{transrate_{i1,i2}} + setupmesg_{i1}$$

- $speed_i$: rychlost zpracování strojem i
- $setuptask_i$: (inicializační) doba na nastartování úlohy na stroji i
- Doba provádění p_{ij} úlohy j na stroji i :

$$p_{ij} = \frac{p_j}{speed_i} + setuptask_i$$

- **Objektivní funkce (performance measure)**
 - minimalizace makespan (čas dokončení poslední úlohy)
 - do makespan se započítává:
 - doba provádění + komunikační zpoždění

Plánování seznamem (list scheduling)

Plánování seznamem = jednoduchý efektivní algoritmus

- založeno na seřazení úloh v **prioritní frontě**
- složitost algoritmu závisí na výpočtu
 - priority určující pořadí spuštění úloh
 - metody výběru stroje (pro danou úlohu)

Používán i při plánování úloh bez precedencí, kde priorita dána

- pořadím přicházejících úloh
- úrovní paralelismu
- délkou trvání
- vlastníkem úlohy
- ...

Algoritmus parametrizován funkcemi

- výpočet priority P
- výběr stroje M

Plánování seznamem(P, M)

- 1 každé úloze j přiřazena priorita pomocí $P(j)$
prioritní fronta inicializována úlohami bez předchůdců
úlohy ve frontě řazeny v klesajícím pořadí dle priority
- 2 pokud existuje volný stroj a fronta je neprázdná, prováděj:
 - 1 jakmile provedeni všichni přímí předchůdci nějaké úlohy, tak je úloha přidána dle priority do fronty
 - 2 odebrána první úloha z fronty
 - 3 volný procesor je vybrán pro spuštění úlohy j pomocí $M(j)$

Algoritmus plánování seznamem a aktuální čas t

Při plánování seznamem je nutné pracovat s **aktuálním časem t**

- rozšíříme základní algoritmus tak, aby byla práce s časem zřejmá

Plánování seznamem včetně času (P, M)

- inicializace aktuálního času $t = 0$
- každé úloze j přiřazena priorita pomocí $P(j)$
prioritní fronta inicializována úlohami bez předchůdců
úlohy ve frontě řazeny v klesajícím pořadí dle priority
- pokud existuje volný stroj a fronta je neprázdná, prováděj:
 - $t =$ nejdřívejší čas, kdy je nějaký stroj volný
 - jakmile provedeni všichni přímí předchůdci j_1, \dots, j_k nějaké úlohy j_0 , tj. $\forall l = 1..k : C_l \leq t$
tak je úloha j_0 přidána dle priority $P(j_0)$ do fronty
 - odebrána první úloha (j) z fronty
 - volný procesor je vybrán pro spuštění úlohy j pomocí $M(j)$
startovní čas S_j úlohy j je t

Priority pro plánování seznamem

Délka cesty přes uzly $j_1, j_2 \dots j_n$:

$$w1 \sum_{j=1}^n p_j + w2 \sum_{j=1}^{n-1} size_{j,j+1}$$

$w1, w2$: koeficienty určující poměr mezi dobou trvání a velikostí přenesených dat

Počáteční uzly *Begin*: uzly bez předchůdců

Koncové uzly *End*: uzly bez následníků

Možnosti pro výpočet priority $P(j_0)$ uzlu j_0

- úroveň (level) bude dále používána pro výpočet priority
délka nejdelší cesty z uzlu j_0 do koncového uzlu

$$P(j_0) = \max_{\forall \text{cesty } (j_0, j_1, \dots, j_k): j_k \in \text{End}} w1 \sum_{j=0}^k p_j + w2 \sum_{j=0}^{k-1} size_{j,j+1}$$

- ko-úroveň (co-level)
délka nejdelší cesty z počátečního uzlu do uzlu j_0

Zanedbání doby na komunikaci:

- pokud máme $j_1 \rightarrow j_2$ a úlohy j_1 a j_2 jsou prováděny na stejném stroji, tak dobu na komunikaci zanedbáme

Značení: koncový čas C_{ij} úlohy j při zpracování na stroji i

Výběr stroje $M(j)$ pro úlohu j :

- pro provádění úlohy jsou vybírány pouze volné stroje
- pro provádění úlohy j je vybrán volný stroj i , na kterém bude úloha j dokončena nejdříve

$$C_j = \min_{\forall i : \text{volny}(i)} C_{ij}$$

- v případě více možností je vybrán stroj s nejmenším indexem
- pozor: při výpočtu koncového času úloh, je třeba brát v úvahu, zda na některém stroji nedojde k zanedbání komunikace

Zjednodušení modelu plánování

Předpokládejme, že

- doba na posílání dat z úlohy j_1 na úlohu j_2 nezávisí na tom, mezi kterými stroji budeme data posílat a odpovídá přímo $size_{j_1,j_2}$

$$c(j_1, j_2, i_1, i_2) = c(j_1, j_2) = size_{j_1, j_2}$$

- $w_1 = w_2 = 1$
- všechny stroje i mají stejnou jednotkovou rychlost

$$speed_i = 1$$

- doba na nastartování úlohy $setuptask_i$ na stroji i je zanedbatelná

⇒ Doba provádění úlohy j není závislá na stroji, tj.

$$p_{ij} = \frac{p_j}{speed_i} + setuptask_i = p_j$$

budeme tedy pro značení doby trvání úlohy j používat pouze p_j

⇒ Úlohy lze plánovat přímo podle dle zadané p_j a $size_{j_1,j_2}$

Koncový čas C_{ij} úlohy j při zpracování na volném stroji i

C_{ij} spočítán na základě

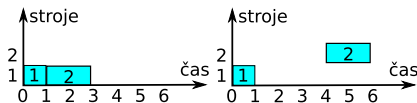
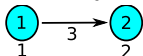
- aktuálního času t (čas, ve kterém úlohu j odebíráme z fronty)
- doby provádění p_j
- komunikace s předchůdci $j1$ ($j1 \rightarrow j$), kteří nebyli prováděni na stroji i

$$\text{pro } \text{volny}(i) : C_{ij} = t + p_j + \sum_{\substack{\forall j1 : j1 \rightarrow j, \\ j1 \text{ prováděna na stroji } i1, i1 \neq i}} \text{size}_{j,j1}$$

Tj. uvažujeme možné zanedbání komunikace

- úloha je ve frontě až v okamžiku, když jsou všichni předchůdci dopočítáni tj. známe jejich stroj a víme, zda dojde k zanedbání komunikace

Příklad: $t = 1$



se zanedbáním komunikace

(na stejném stroji)

bez zanedbání komunikace

(na různých strojích)

Výběr stroje: algoritmus (shrnutí)

Výběr stroje $M(j)$ pro úlohu j v čase t

① $C_j = \text{MaxNumber}$

② REPEAT

① $i =$ nejmenší index dosud neprozkoumaného volného stroje

② spočítáme koncový čas C_{ij} úlohy j na stroji i

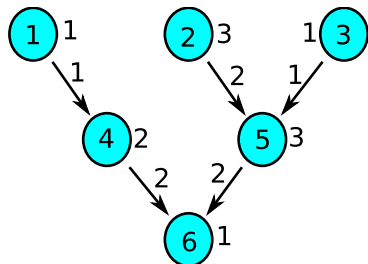
$$C_{ij} = t + p_j + \sum_{\substack{\forall j_1 : j_1 \rightarrow j, \\ j_1 \text{ prováděna na stroji } i_1, i_1 \neq i}} \text{size}_{j,j_1}$$

③ pokud je koncový čas na stroji i lepší,
tak je i novým kandidátem na zpracování úlohy j
IF $C_{ij} < C_j$ THEN $M(j) = i$

UNTIL jsme neprošli všechny volné stroje

Plánování seznamem: příklad

Plánování 6 úloh zadaných grafem na 2 stroje se stejnou rychlostí

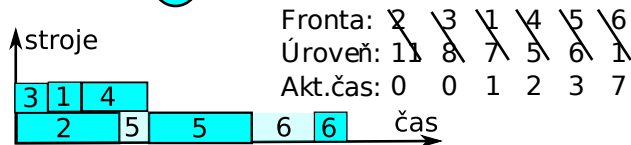


komunikace zanedbána:

1->4

2->5

5->6



- tmavší obdélníky v rozvrhu znázorňují dobu, kdy úloha běží
- světlejší obdélníky odpovídají době na komunikaci pro danou úlohu
 - pro 5: komunikace 3 → 5, pro 6: komunikace 4 → 6

Vyřešte následující problém plánování s komunikací a s precedencemi pomocí plánování seznamem:

- jsou dány 2 stroje se stejnou rychlostí
- je dáno 6 úloh s dobou trvání
 $p_1 = 1, p_2 = 1, p_3 = 2, p_4 = 1, p_5 = 2, p_6 = 1$
- jsou dány precedence: $1 \rightarrow 2, 2 \rightarrow 4, 4 \rightarrow 6, 1 \rightarrow 3, 3 \rightarrow 5, 5 \rightarrow 6$
- $size_{i_1, i_2}$ reprezentuje dobu na přenos dat a je zadán takto:
 $size_{1,2} = 1, size_{2,4} = 3, size_{4,6} = 1,$
 $size_{1,3} = 2, size_{3,5} = 4, size_{5,6} = 2$

Výběr úlohy s nejvyšší úrovní = výběr úlohy na kritické cestě

Problémy při použití heuristiky

- snadno může dojít ke **změně kritické cesty**: důsledkem toho, že
 - ⇐ komunikační zpoždění závisí na alokaci úlohy na stroj
 - ⇐ např. při alokaci na stejné stroje se zanedbatelným zpožděním nebo při nestejnoměrné vzdálenosti strojů (a počtu hopů)

Řada heuristik založena na principu (několika) prioritních front

- rozsáhlé modifikace priorit a metod výběru stroje umožňují zachytit různé charakteristiky systému
- např. **produkční plánovací systémy** pro plánování úloh na počítačích PBSP_{ro}, SGE

Heuristiky mapování (mapping heuristics)

- Rozšíření metod plánování seznamem
- Pro návrh heuristik uvažovány další reálné parametry:
 - topologie sítě, rychlost procesoru, přenosová rychlost, ...
- Doba provádění úlohy i na stroji j

$$p_{ij} = \frac{p_i}{speed_j} + setup_{task_j}$$

- Komunikační zpoždění

$$c(j1, j2, i1, i2) = \left(\frac{size_{j1, j2}}{transrate} + setup_{msg} \right) \times hops_{i1, i2} + contdelay_{i1, i2}$$

- předpokládán konstantní přenosová rychlost $transrate$ a inicializační doba na posláni zprávy $setup_{msg}$
- $hops_{i1, i2}$: počet hopů mezi stroji $i1$ a $i2$, (délka nejkratší cesty mezi stroji při jednotkovém ohodnocení hran)
- $contdelay_{i1, i2}$: zpoždění dané zatížením linky mezi $i1$ a $i2$ (*contention delay*)

Heuristiky mapování (mapping heuristics)

- Konstruována a udržována **směrovací tabulka** obsahující
 - odhadované hodnoty p_{ij} a $c(j_1, j_2, i_1, i_2)$
 - pro každý procesor
 - počet hopů, preferovaná linka pro daný cíl, zpoždění dané zatížením
- **Koncový čas úlohy na stroji** spočítán z
 - rychlost procesoru, přenosová rychlost, vybraná linka, počet hopů, zpoždění dané zatížením linky

Jedná se o plánování seznamem, tj.

- **Úlohy plánovány podle nejvyšší úrovně**, v případě nejednoznačnosti vybrána úloha s větším množstvím přímých následníků
- Jakmile jsou dokončeni všichni předchůdci úlohy, tak je **úloha naplánována na stroj, kde nejdříve skončí**

Plánování se shlukovacími heuristikami (clustering heuristics)

- Plánování řeší
 - 1 přiřazení úloh na stroje
 - 2 seřazení úloh na stroji / umístění úloh v čase
- **Shlukovací heuristiky**: primárně řeší přiřazení (alokaci) úloh na stroje
- Následné naplánování úloh v čase na vybraný stroj je realizováno jednoduchou aplikací plánování seznamem
- **Shluk**: množina úloh, která bude prováděna na stejném stroji

Princip plánování založeného na shlukování

- 1 Shlukování úloh na nelimitovaný počet plně propojených procesorů
- 2 Namapování shluků a jejich úloh na daný počet procesorů (m) provedením následujících operací
 - 1 spojování shluků: pokud je počet shluků vyšší než m
 - 2 fyzické mapování shluků: přiřazení shluků na procesory tak, aby byla minimalizována komunikace (na této úrovni uvažována reálná konektivita mezi stroji)
 - 3 uspořádání úloh: úlohy jsou na stroji prováděny tak, by byly splněny závislosti mezi úlohami, např. pomocí zjednodušeného plánování seznamem na jeden stroj

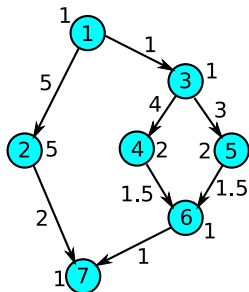
- Shlukování
 - bez backtrackingu (abychom se vyhnuli vysoké složitosti), tj. jakmile jsou jednou shluky spojeny nelze je zpětně rozpojit
- Iniciálně: jedna úloha = jeden shluk
- Typický krok heuristiky shlukování
 - spojení shluků + **vynulování hrany**, která je spojuje
 - vynulování hrany: úlohy prováděny na stejném procesoru, kde je čas na komunikaci zanedbatelný
- Kritérium pro výběr hrany na nulování redukce tzv. **paralelního času rozvrhu**

Naplánovaný graf úloh

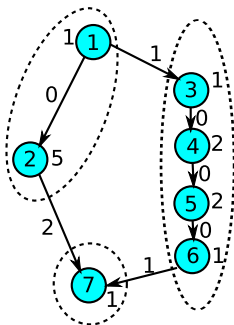
Naplánovaný graf úloh

- graf úloh zahrnující aktuální **vynulování hran ve shluku**
- úlohy v jednom shluku jsou (jako na procesoru)
seřazeny dle nejvyšší úrovně

původní graf úloh



naplánovaný graf úloh se 3 shluky

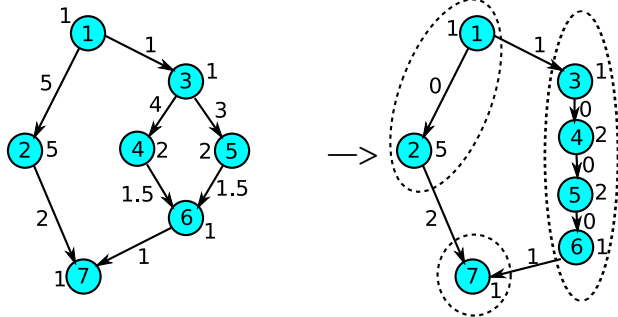


- Připomenutí: délka cesty přes uzly $j_1, j_2 \dots j_n$:

$$w_1 \sum_{j=1}^n p_j + w_2 \sum_{j=1}^{n-1} size_{j,j+1}$$

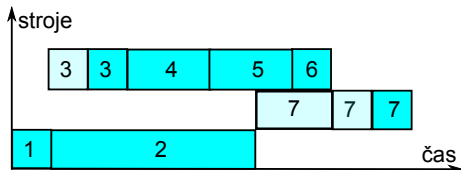
- **Dominantní posloupnost:**
nejdelší cesta v naplánovaném grafu úloh
- **Paralelní čas rozvrhu:**
doba nutná na provedení dominantní posloupnosti
- Pokud $w_1 = w_2 = 1$ a $size_{j_1,j_2}$ reprezentuje dobu na přenos mezi j_1, j_2
 \Rightarrow délka cesty odpovídá době na provedení úloh na cestě
 \Rightarrow paralelní čas rozvrhu = délka cesty dominantní posloupnosti

Dominantní posloupnost: příklad



Dominantní posloupnost 1,3,4,5,6,7 s délkou 10, tj. paralelní čas rozvrhu je 10

Odpovídající rozvrh na 3 strojích:



POZOR: Přenos dat z úlohy 2 na úlohu 7 může být zahájen hned, jakmile skončí úloha 2 (víme, na kterém stroji úloha 7 poběží)

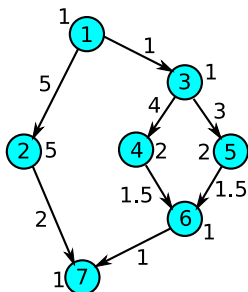
Shlukování

- 1 Inicializace: všechny hrany označeny jako nevyzkoušené
každá úloha tvoří jeden shluk
 - 2 Seřazení všech hran (j_1, j_2) v grafu úloh v klesajícím pořadí
dle komunikační ceny ($size_{j_1, j_2}$)
 - 3 REPEAT
 - 1 vynulování největší nevyzkoušené hrany v seřazeném seznamu,
pokud nevzroste paralelní čas
 - 2 hrana je označena jako vyzkoušená
 - 3 když jsou spojeny dva shluky, úlohy jsou uspořádány dle nejvyšší úrovně
- UNTIL všechny hrany jsou vyzkoušené

Komentář: *podobně jako existují různé metody plánování seznamem, tak existují různé varianty shlukování*

Shlukování: příklad

Problém:

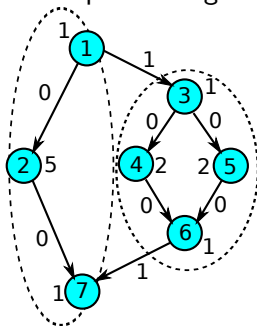


Postup:

seřazení hran, postupné zkoušení na vynulování a spojování shluků:

12 (vynulována, shluk 12), 34 (vynulována, shluk 34), 35 (vynulována, shluk 345), 27 (vynulována, shluk 127), 46 (vynulována, shluk 3456), 56 (vynulována), 13 (nevynulována, paralelní čas by z 10 pro dominantní cestu 1,3,4,5,6,7 vzrostl na 13 s dominantní cestou 1,2,3,4,5,6,7), 67 (nevynulována stejně jako 13)

Řešení pomocí algoritmu:



Pokud je shluků méně než strojů

- každý shluk přiřadíme na jeden stroj (zbývající stroje zůstanou volné)

Pokud je shluků stejně jako strojů

- shluk = stroj

Pokud máme shluků více je nutné aplikovat **spojování shluků**

- lze využít **problému plnění košů** (bin packing problem) a heuristik pro jeho řešení
 - plnění předmětů do košů tak, aby byly všechny koše rovnoměrně naplněny
 - 1 stroj = 1 koš
 - 1 shluk = 1 předmět dané velikosti
 - velikost shluku $S = \sum_{j \in S} p_j$
 - cíl: součet velikostí předmětů ve všech koších je přibližně stejný, tj. stroje jsou přibližně stejně vytížené

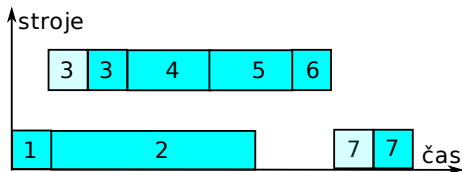
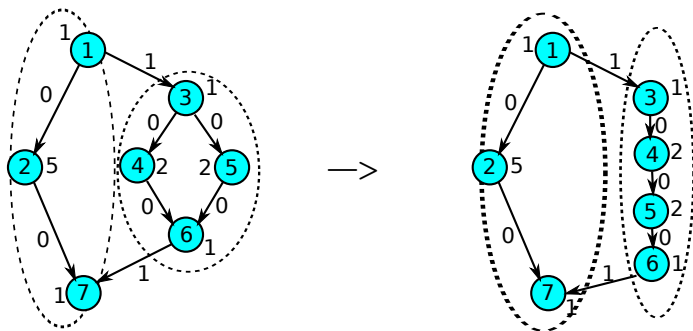
Plánování pomocí shlukovací heuristiky:

- 1 **shlukování**
- 2 **spojování shluků**, pokud je shluků více než strojů
- 3 **fyzické mapování** spojených shluků na stroje
 - obecně: přiřazení shluků na procesory tak, aby byla minimalizována komunikace
- 4 **uspořádání úloh na každém stroji**

např. pomocí jednoduchého plánování seznamem

 - není nutno brát v úvahu $M(j)$, stroj už je předem vybrán
 - stačí uvažovat prioritu $P(j)$ /úroveň a seřadit úlohy dle jejich úrovně
 - přenos dat k následníkům úlohy je zahájen hned po jejím dokončení (víme, na kterém stroji budou úlohy spuštěny)
 - úlohy pak mohou být spuštěny na stroji, jakmile jsou dokončeni všichni jejich předchůdci (a přenesena data)

Uspořádání úloh na stroji: příklad



Vyřešte cvičení z kapitoly plánování seznamem pomocí algoritmu shlukování. Následně uspořádejte úlohy na stroji (použijte stejný počet strojů jako máte shluků) a zkonstruujte výsledný rozvrh.

Je v řešení nějaký rozdíl?

(náповěda: ano, u shlukovacích heuristik víme, na kterém stroji bude úloha zpracovávána předem, a proto je možné zahájit některé komunikace dříve, a tedy rozvrh skončí dříve)