MPMD

- Viz 2. přednáška
- Ve všech uzlech není stejný program
- Každý proces má svoje data

SPMD

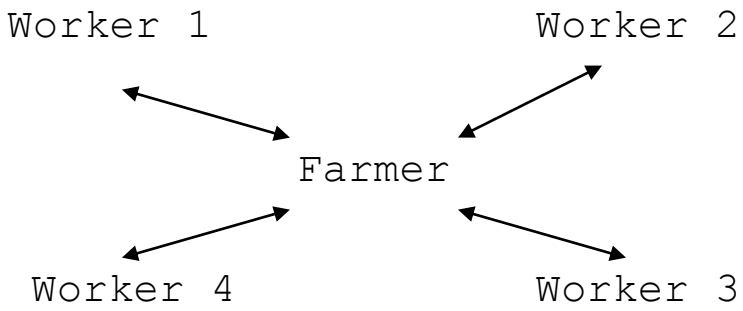
- Viz 3. přednáška
- Do všech uzlů se zavede stejný program
- Do uzlů se zavedou specifická data procesů
- Každý proces dokáže zjistit svoje ID a z něho určí sousedy a svůj díl dat
- Jeden z procesů je řídící
  - Obvykle první vytvořený
    - Mívá ID 0 – závisí na sw
  - Zpracovává dílčí mezivýsledky

- V homogenním distribuovaném prostředí se zjistí celkový objem dat, počet spuštěných procesů a práce se přidělí najednou
  - Např. cluster z identických stanic s MPI

- V heterogenním prostředí je lepší využít dynamické přidělování práce
  - Uzly nemusejí být stejně výkonné
  - Ale i v případě, že na uzly nejsou využívány jednouživatelsky
  - Např. model farmer-workers, kdy farmáři udělíme „čestný titul“ identický program:-)
    - Technicky vzato, program může být identický, pouze řídící proces bude vykonávat i řídící část

Farmer-Workers

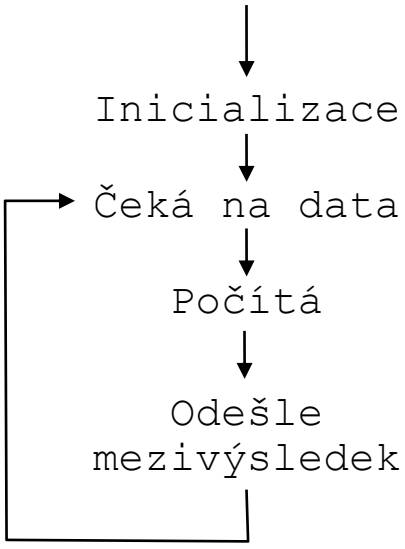
- N procesů
- 1 řídící – farmer
- N-1 dělníků



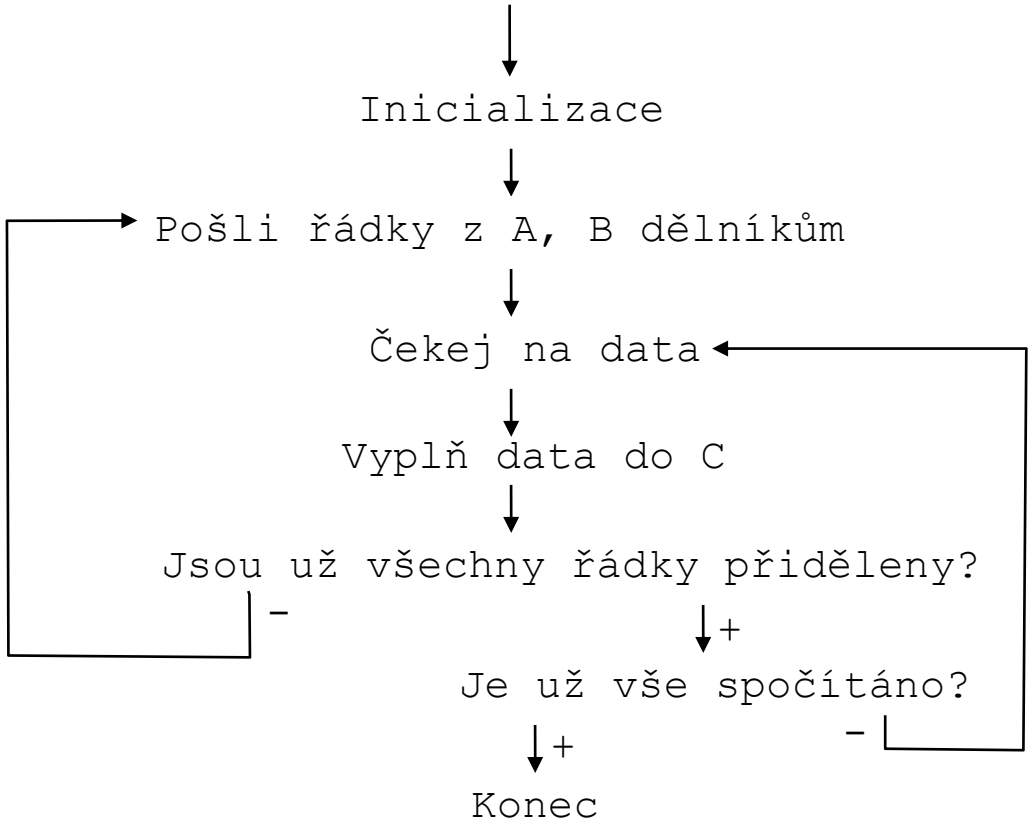
- Virtuální topologie je hvězda
- Současně může být zasíláno 2*(N-1) zpráv – plný duplex

- Součet matic
  - Farmer má A, B
  - Cílem je součet v C

- Worker



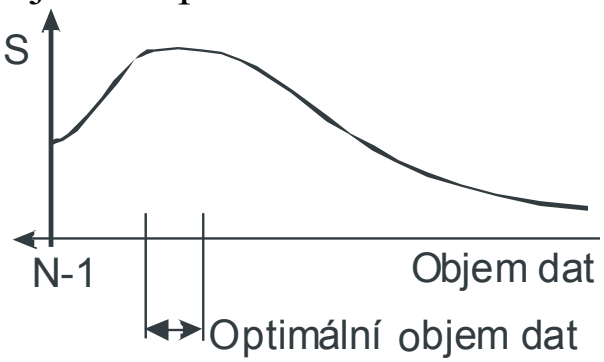
- Farmer



- Urychlení
  - Zanedbáme-li dobu vysílání a příjmu zprávy, pak $S \sim N - 1$, tj. přibližně lineární

  - Hodí se tehdy, když
    - Datový objem zpráv je malý
    - Výpočet je v porovnání s objemem zprávy náročný
    - Např. nemá smysl posílat pole integerů k součtu na jiný uzel, protože u dnešních i86/64 je doba trvání MOV a ADD zhruba stejná
      - Zprávu je třeba připravit do nějakého bufferu, než se odešle a k tomu se v nějakém bodě překopíruje instrukcemi MOV, kterými se z něho také vybalí

- V heterogenním prostředí se realizuje automatický load-balancing (viz pozdější přednáška)
- Urychlení pak závisí na velikosti objemu dat ke zpracování jedním procesem



- Příliš malé objemy dat nevedou k urychlení
- Příliš velké objemy dat jsou sice rychlejší než příliš malé, ale zase se zbytečně čeká na procesy, které z nějakého důvodu počítaly pomaleji např. vlivem hw, nebo jiného sw, či režie OS, …

- Ideální velikost posílaných dat?
  - Závisí na výpočetním výkonu uzlů
  - Měla by uzel na zaměstnat tak dlouho, aby bylo možné úkolovat uzly v době, kdy ostatní počítají
    - V první „vlně“ přidělit náhodné velikosti dat od jednoho až dvou násobků objemu zprávy
      - Tím se na nějakou dobu zabrání konvergenci, kdy bude komunikační linka používána všemi procesy
        - Eliminace komunikačního zpoždění překrytím výpočtem
          - Kromě neblokujících komunikačních operací
  - Data by se měla spočítat v nějakém přiměřeném čase, se na poslední proces nečekalo „celou věčnost“

MPSD

- Viz 1. přednáška
- Step-locked
- „pásová“ výroba

- Části dat by měly být stejně velké ne podle objemu dat, ale výpočetně
- Problémem může být kapacita přenosových kanálů
  - Objem dat může být příliš velký a tak uzel může nějakou dobu čekat na data
  - Ideálně se v i-tém kroku
    - Počítají data
    - Data z i-1 kroku se posílají dalšímu uzlu v řadě
    - Přijímají se data, která se budou počítat v i+1 kroku
    - => překrytí doby komunikace dobou výpočtu
      - => eliminace komunikačního zpoždění, pokud se data déle počítají, než přijímají
        - Pokud ne, zmenšit objem posílaných dat

- N – počet úkolů
- Pošleme-li objem vstupních dat k nekonečnu, pak je urychlení N