

IB109 Návrh a implementace paralelních systémů

Kolektivní komunikační primitiva

Jiří Barnat

Komunikace (interakce)

- Předávání informací mezi jednotlivými procesy

Parametry komunikace

- **Latence** — zpoždění související se započítáním vlastní komunikace
- **Šířka pásma** (bandwidth) — maximální množství dat přenesených za jednotku času
- **Objem** — množství předávaných dat

Cena komunikace

- t_s – latence, t_w – šířka pásma, m – objem
- $t_s + m * t_w$

Příklady

- Za jak dlouho napustím hrníček vodou pomocí 2km dlouhé zahradní hadice?
- Při konstantním čtení z paměti trvá získání kódu instrukce a příslušných operandů z paměti v průměru 5ns. Jaká je nejvyšší reálná rychlost vykonávání kódu uloženého v paměti 4GHz procesorem?

$$\left[1/(5 * 10^{-9}) = 0.2 * 10^9 = 200 \text{ MHz} \right]$$

Pozorování

- Interakce jednotlivých úloh/procesů je nevyhnutelná
- Interakce způsobuje prodlevy ve výpočtu
- Režie související s interakcí by neměla být důvodem pro neefektivitu paralelního zpracování

Závěr

- Komunikační primitiva musí být co nejefektivnější

V této přednášce

- Popis různých typů komunikačních operací
- Odvození ceny pro jednotlivé topologie

Topologie

- Fyzická/**logická** struktura komunikačních kanálů mezi jednotlivými participanty komunikace.

Vlastnosti komunikační sítě

- Průměr (délka maximální nejkratší cesty)
- Konektivita (minimální počet disjunktních cest)
- Stupeň (max. počet linek incidenčních s jedním vrcholem)
- Cena
- Výlučnost přístupu (použití jednou úlohou blokuje ostatní)
- Rozšiřitelnost
- Škálovatelnost

Sběrnice

- Sdílené médium
- Propustnost, klesá s počtem uzlů (je třeba cache)
- Důležitost: model sdíleného adresového prostoru

Kliky (úplné síť)

- Privátní neblokující spojení každého s každým
- Cena: počet linek je kvadratický vůči N
- Cena: stupeň každého uzlu je $N - 1$
- Škálovatelné, za podmínky levného zvyšování stupně uzlu
- Důležitost: abstraktní představa sítě, logická struktura

Prsten (kruh, řetěz, 1-rozměrná mřížka)

- Uspořádání na uzlech
- Privátní neblokující komunikace s 2 nejbližšími uzly
- Důležitost: logická struktura komunikace, Pipeline model

Hvězdice

- Spojení přes jeden centrální uzel
- Středový uzel může být úzkým místem
- Lze hierarchicky vrstvit (hvězdice hvězdic)
- Důležitost: Master-Slave model

Hyperkostka

- Spojení n uzlů ve tvaru $\log_2 n$ -rozměrné krychle
- Stupeň uzlu je $\log_2 n$
- Průměr sítě je $\log_2 n$
- Počet linek $O(N \cdot \log(N))$
- Postup konstrukce
 - binární označení uzlů s identifikátory 0 až $2^{\log_2 n} - 1$
 - linka existuje mezi uzly pokud se označení liší v jednom bitu
 - minimální vzdálenost uzlů odpovídá počtu odlišných bitů v označení uzlů

Strom s aktivními vnitřními uzly

- Strom, kde participanty komunikace jsou listy i vnitřní uzly
- Kostra hyperkostky — Strom s maximální hloubkou $\log_2 n$
- Důležitost: Optimální šíření informací

Jiné topologie

- Z hlediska logického návrhu paralelní aplikace nezajímavé.

Příklady

- Obecné stromy
- Přepojované sítě
- Vícevrstvé sítě (ω -sítě)
- Mřížky
- Cyklické mřížky (torrus)

Jeden na všechny a všichni na jednoho

One-To-All Vysílání (OTA)

- Úloha posílá několika/všem ostatním identická data
- Ve výsledku je p kopií originální zprávy v lokálních adresových prostorech adresátů
- Změna struktury dat: $m \mapsto p * m$ (m -je velikost zprávy)

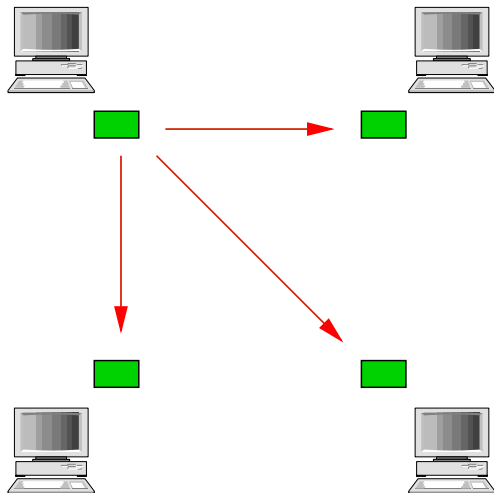
All-To-One Redukce (ATO)

- Duální operace k “One-To-All”
- Několik/všechny úlohy posílají data jedné úloze
- Data se kombinují pomocí zvoleného asociativního operátoru
- Ve výsledku je jedna kopie v adresovém prostoru cílové úlohy
- $m_1 \otimes m_2 \dots \otimes m_p \rightsquigarrow m$
- Změna struktury dat: $m * p \mapsto m$

OTA: Změna struktury dat



OTA: Změna struktury dat



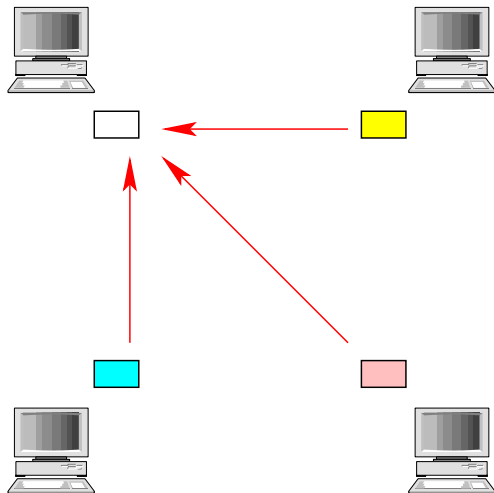
OTA: Změna struktury dat



ATO: Změna struktury dat



ATO: Změna struktury dat



ATO: Změna struktury dat



Naivní způsob One-To-All pro p procesů

- Poslat $p - 1$ zpráv postupně ostatním procesům
- Úzké místo: odesílatel
- Síť je nevyužitá, komunikuje pouze jedna dvojice procesů

Technika rekurzivního zdvojení (připomenutí)

- Nejprve první proces pošle zprávu jinému procesu
- Poté oba procesy pošlou zprávu jiné dvojici
- Poté čtveřice procesů pošle zprávu jiné čtveřici
- ...
- První proces pošle nejvýše $\log(p)$ zpráv
- Souběh zpráv na linkách sítě
- Optimální vzdálenosti adresátů pro jednotlivá kola jsou $p/2$, $p/4$, $p/8$, $p/16$, ...

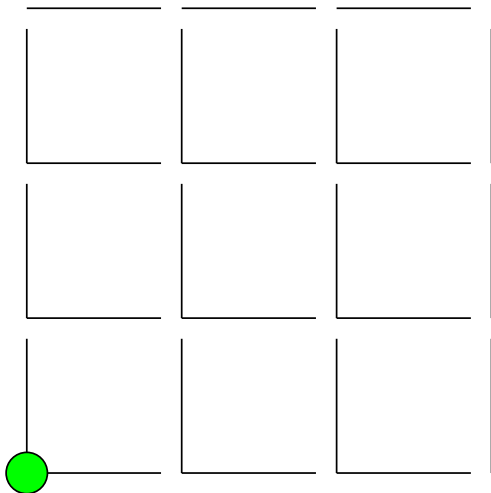
One-To-All ve 2-rozměrné síti o p uzlech

- Lze chápat jako \sqrt{p} řetízků o \sqrt{p} uzlech
- V první fázi se propaguje informace do všech řetízků
- V druhé fázi se souběžně propaguje informace v jednotlivých řetížcích
- Celková cena: $2(\log(\sqrt{p}))$ sekvenčně provedených operací

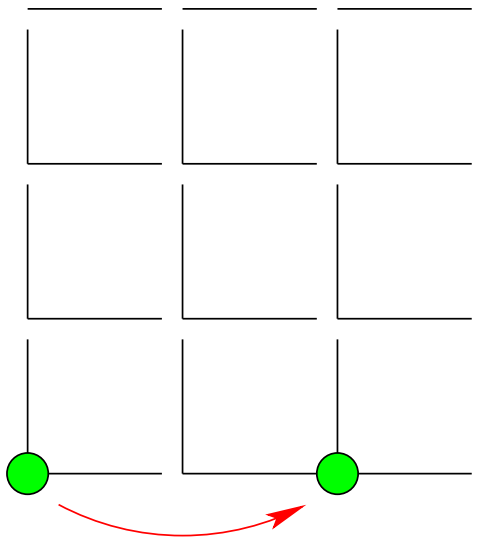
One-To-All v d -rozměrné síti

- Stejný princip, velikost v jednom rozměru je $p^{1/d}$
- Celková cena: $d(\log(p^{1/d}))$

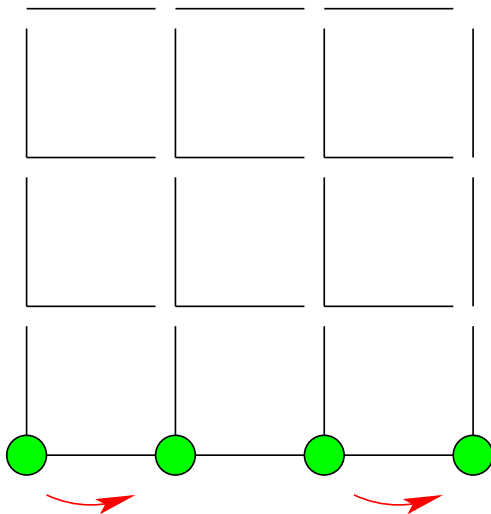
OTA: Pro topologii 2-dimenzionální mřížka

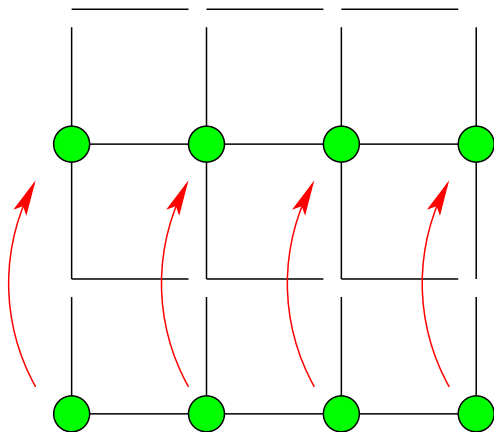


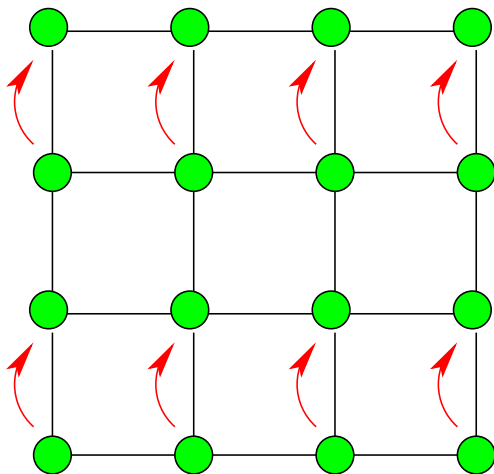
OTA: Pro topologii 2-dimenzionální mřížka

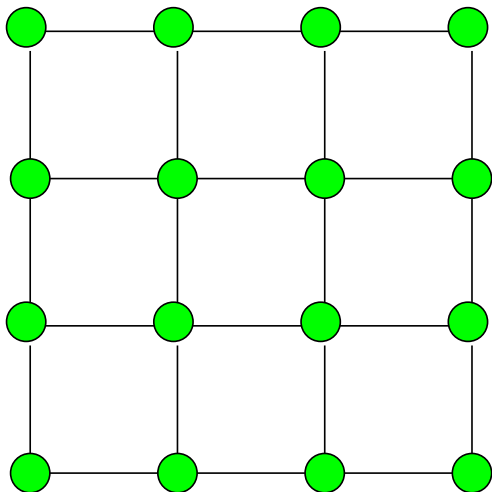


OTA: Pro topologii 2-dimenzionální mřížka









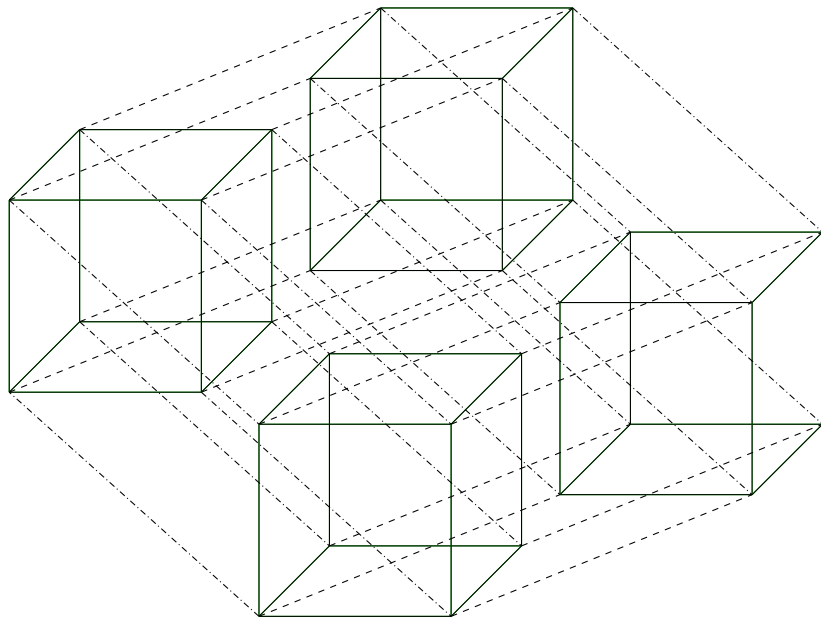
Hyperkostka s 2^d vrcholy

- Aplikuje se algoritmus pro síť ve tvaru mřížky
- d -dimenzionální síť s hloubkou sítě 2
- Každá fáze proběhne v konstantním čase (poslání 1 zprávy)
- Nenastává souběžný přístup na žádnou z komunikačních linek
- Celková cena: d

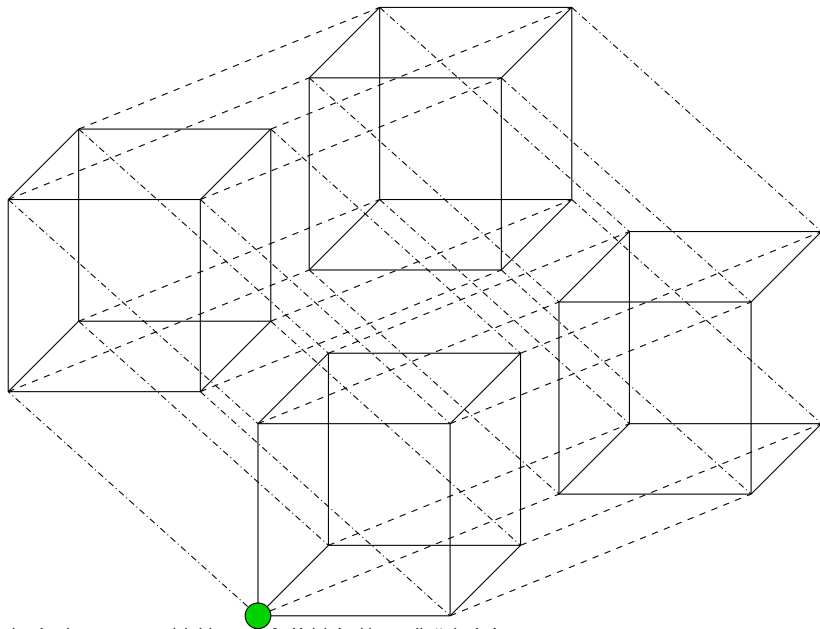
Příklad

- Jak proběhne One-To-All na hyperkostce s dimenzí 5?

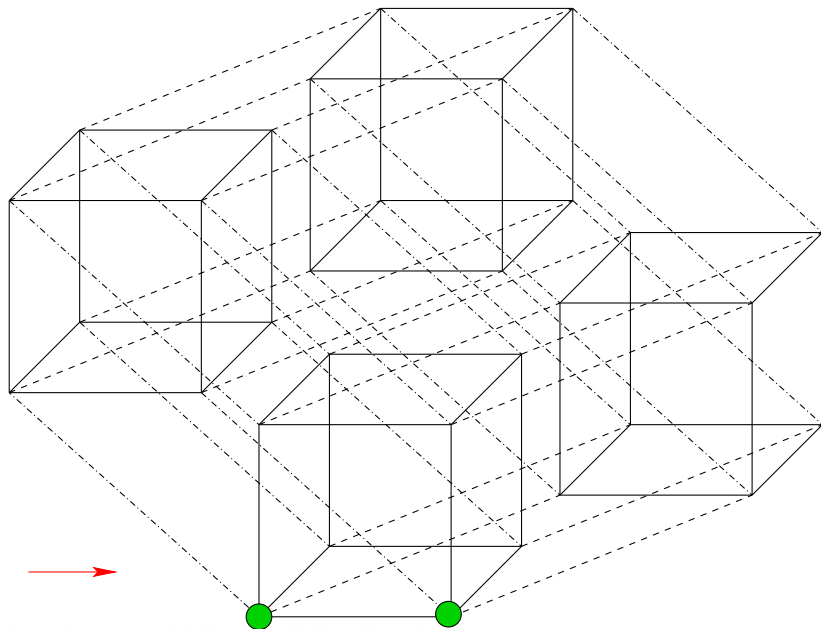
OTA: Pro hyperkostku o dimenzi 5



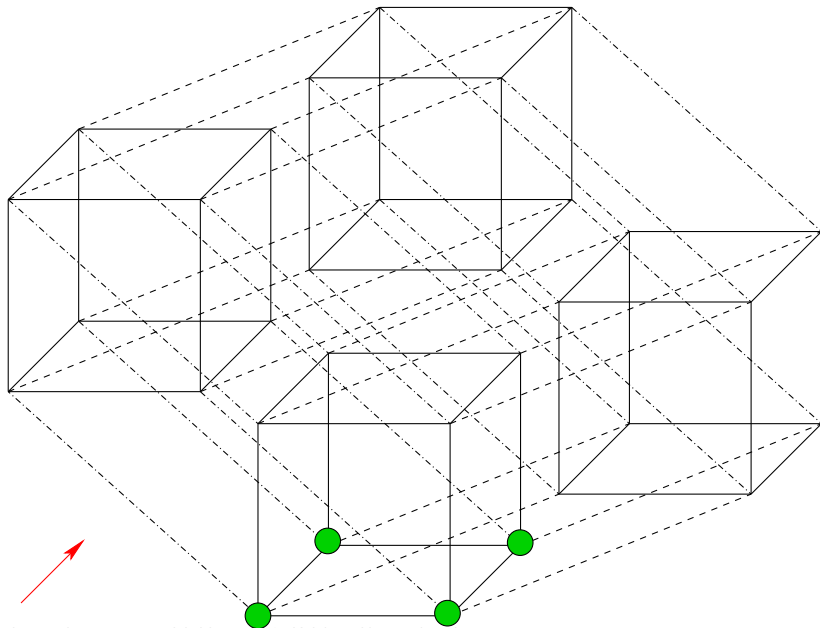
OTA: Pro hyperkostku o dimenzi 5



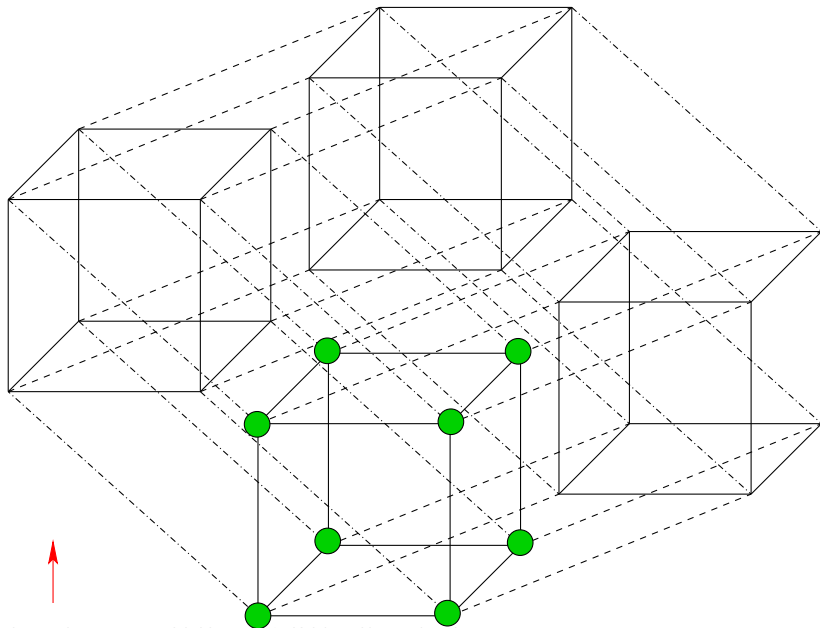
OTA: Pro hyperkostku o dimenzi 5



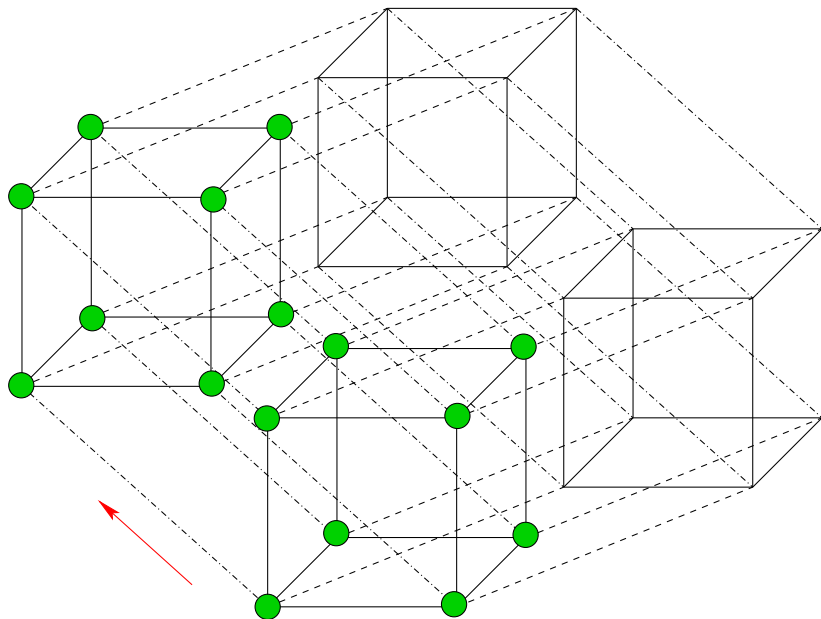
OTA: Pro hyperkostku o dimenzi 5



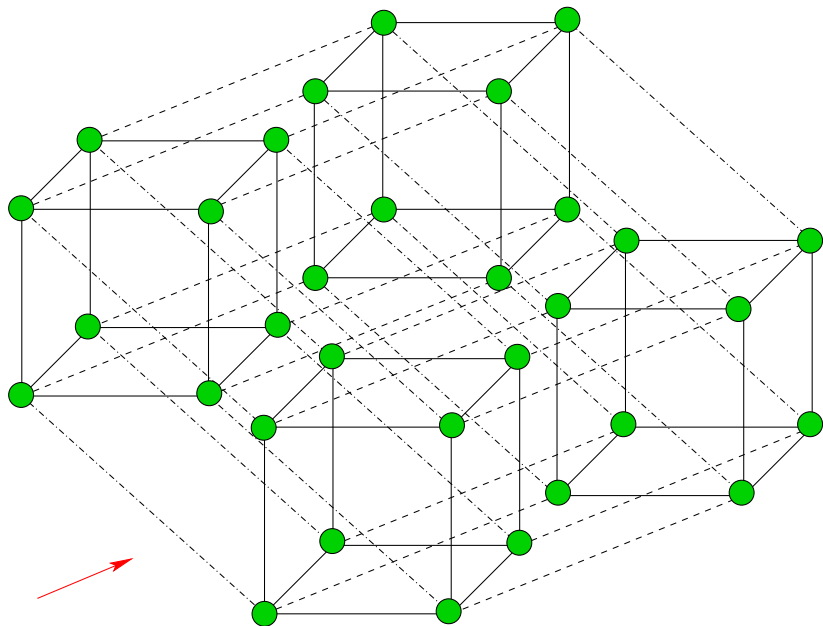
OTA: Pro hyperkostku o dimenzi 5



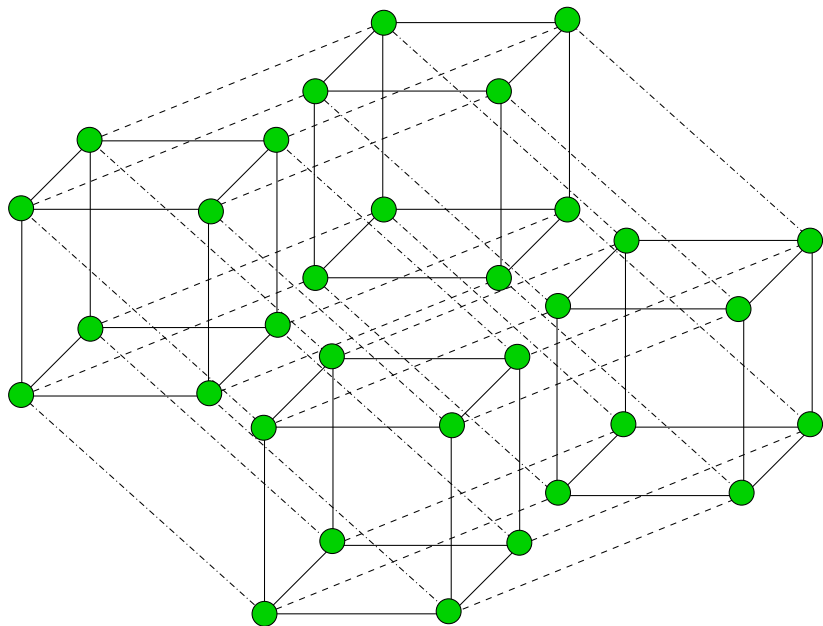
OTA: Pro hyperkostku o dimenzi 5



OTA: Pro hyperkostku o dimenzi 5



OTA: Pro hyperkostku o dimenzi 5

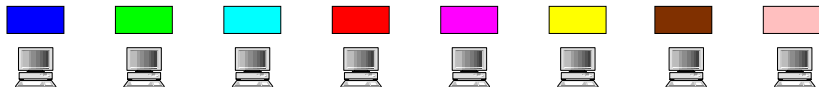


All-To-One Redukce pro p procesů

- Probíhá duálně k operaci One-To-All

Příklad: ATO pro topologie prsten či řetěz

- Prvně procesy s lichým ID pošlou zprávu procesům s ID o jedna menším, kde se zprávy zkombinují s hodnotou, kterou chtějí vyslat procesy se sudým ID
- Následně proběhne kombinace informací sousedních procesů se sudým ID na procesech jejichž ID je násobkem čtyř
- ...

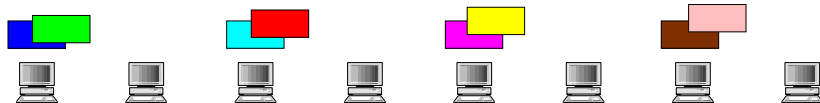


All-To-One Redukce pro p procesů

- Probíhá duálně k operaci One-To-All

Příklad: ATO pro topologie prsten či řetěz

- Prvně procesy s lichým ID pošlou zprávu procesům s ID o jedna menším, kde se zprávy zkombinují s hodnotou, kterou chtějí vyslat procesy se sudým ID
- Následně proběhne kombinace informací sousedních procesů se sudým ID na procesech jejichž ID je násobkem čtyř
- ...



All-To-One Redukce pro p procesů

- Probíhá duálně k operaci One-To-All

Příklad: ATO pro topologie prsten či řetěz

- Prvně procesy s lichým ID pošlou zprávu procesům s ID o jedna menším, kde se zprávy zkombinují s hodnotou, kterou chtějí vyslat procesy se sudým ID
- Následně proběhne kombinace informací sousedních procesů se sudým ID na procesech jejichž ID je násobkem čtyř
- ...



All-To-One Redukce pro p procesů

- Probíhá duálně k operaci One-To-All

Příklad: ATO pro topologie prsten či řetěz

- Prvně procesy s lichým ID pošlou zprávu procesům s ID o jedna menším, kde se zprávy zkombinují s hodnotou, kterou chtějí vyslat procesy se sudým ID
- Následně proběhne kombinace informací sousedních procesů se sudým ID na procesech jejichž ID je násobkem čtyř
- ...



All-To-One Redukce pro p procesů

- Probíhá duálně k operaci One-To-All

Příklad: ATO pro topologie prsten či řetěz

- Prvně procesy s lichým ID pošlou zprávu procesům s ID o jedna menším, kde se zprávy zkombinují s hodnotou, kterou chtějí vyslat procesy se sudým ID
- Následně proběhne kombinace informací sousedních procesů se sudým ID na procesech jejichž ID je násobkem čtyř
- ...



Pozorování

- One-To-All procedury jsou si podobné
- Jdou nahradit univerzální procedurou

Univerzální algoritmy OTA vysílání a ATO Redukce

- Předpokládají 2^d uzlů (hyperkostka)
- Každý uzel identifikován bitovým vektorem
- *AND* a *XOR* bitové operace

Cena

- $d(t_s + mt_w)$

```
(1) proc ONE-TO-ALL( $d, id, X$ )
(2)    $mask := 2^d - 1$ 
(3)   for  $i := d - 1$  downto 0
(4)      $mask := mask \text{ XOR } 2^i$ 
(5)     if  $(id \text{ AND } mask) = 0$ 
(6)       then if  $(id \text{ AND } 2^i) = 0$ 
(7)         then  $msg\_destination := id \text{ XOR } 2^i$ 
(8)           send  $X$  to  $msg\_destination$ 
(9)         else  $msg\_source := id \text{ XOR } 2^i$ 
(10)          receive  $X$  from  $msg\_source$ 
(11)        fi
(12)     fi
(13)   end
(14) end
```

```
(1) proc GENERAL-ONE-TO-ALL( $d, id, src, X$ )
(2)    $virtid := id \text{ XOR } src$ 
(3)    $mask := 2^d - 1$ 
(4)   for  $i := d - 1$  downto 0
(5)      $mask := mask \text{ XOR } 2^i$ 
(6)     if  $(virtid \text{ AND } mask) = 0$ 
(7)       then if  $(virtid \text{ AND } 2^i) = 0$ 
(8)         then  $virt\_destination := virtid \text{ XOR } 2^i$ 
(9)           send  $X$  to  $(virt\_destination \text{ XOR } src)$ 
(10)        else  $virt\_source := virtid \text{ XOR } 2^i$ 
(11)         receive  $X$  from  $(virt\_source \text{ XOR } src)$ 
(12)        fi
(13)     fi
(14)   end
(15) end
```

```
(1) proc ALL-TO-ONE( $d, id, m, X, sum$ )
(2)   for  $j := 0$  to  $m - 1$  do  $sum[j] := X[j]$  end
(3)    $mask := 0$ 
(4)   for  $i := 0$  to  $d - 1$ 
(5)     if  $(id \text{ AND } mask) = 0$ 
(6)       then if  $(id \text{ AND } 2^i) = 0$ 
(7)         then  $msg\_destination := id \text{ XOR } 2^i$ 
(8)           send  $sum$  to  $msg\_destination$ 
(9)         else  $msg\_source := id \text{ XOR } 2^i$ 
(10)          receive  $X$  from  $msg\_source$ 
(11)          for  $j := 0$  to  $m - 1$ 
(12)             $sum[j] := sum[j] + X[j]$ 
(13)          end
(14)        fi fi
(15)       $mask := mask \text{ XOR } 2^i$ 
(16)    end
(17) end
```

Všichni všem a všichni od všech

All-To-All Vysílání

- Všichni posílají informaci všem
- Každá úloha posílá jednu zprávu všem ostatním úlohám
- Ve výsledku je p kopií p originálních zpráv v lokálních adresových prostorech adresátů
- Změna struktury dat: $p * m \mapsto p * p * m$

All-To-All Redukce

- Všichni posílají informaci všem, příchozí zprávy se kombinují
- Každá úloha má pro každou jinou úlohu jinou zprávu
- Změna struktury dat: $p * p * m \mapsto p * m$

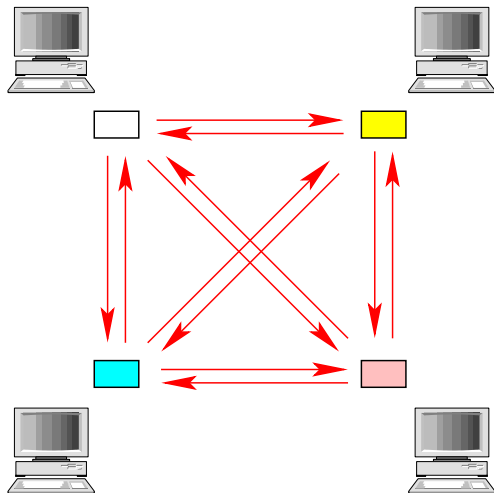
Naivní řešení

- Sekvenční/násobné použití One-To-All procedur
- Neefektivní využití sítě

ATA Vysílání: Změna struktury dat



ATA Vysílání: Změna struktury dat



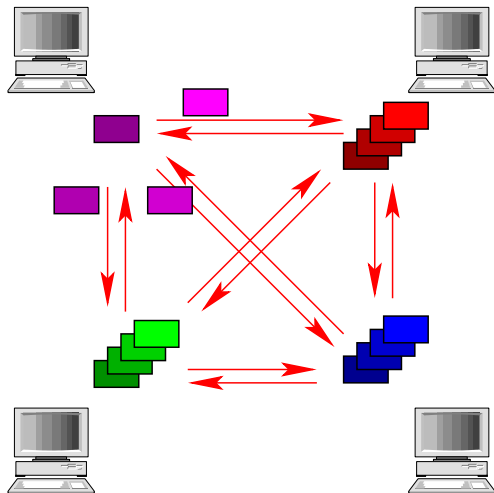
ATA Vysílání: Změna struktury dat



ATA Redukce: Změna struktury dat



ATA Redukce: Změna struktury dat



ATA Redukce: Změna struktury dat



Prsten

- 1. fáze: každý uzel pošle informaci svému sousedovi
- 2. až n -tá fáze: uzly sbírají a přeposílají příchozí zprávy
- Všechny linky jsou po celou dobu operace plně využité
- Optimální algoritmus

Řetěz

- Vysílání zpráv sousedům na obě strany
- Full-duplex linky $\Rightarrow n - 1$ fází
- Half-duplex linky $\Rightarrow 2(n - 1)$ fází

ATA Redukce

- Zprávy po jedné posílány po kruhu tak, že zpráva pro nejbližší uzel je poslána jako první
- Při průchodu uzlem, se přikombinuje lokální zpráva pro adresáta právě procházející zprávy

Mřížka

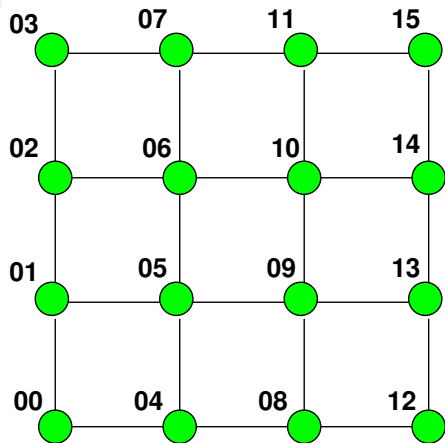
- 2 fáze – \sqrt{p} řetízků o \sqrt{p} uzlech
- Po první fázi uzly mají uzly \sqrt{p} částí z celkového počtu p částí zprávy
- Příklad sítě 4x4, každý posílá své ID každému

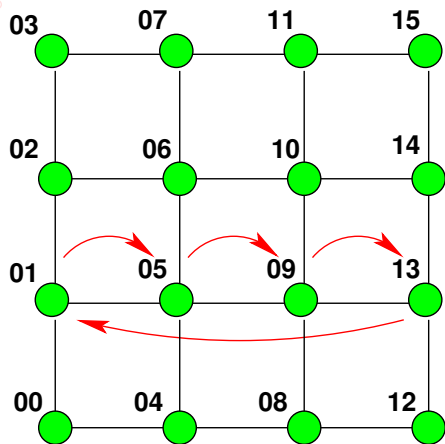
Hyperkostka

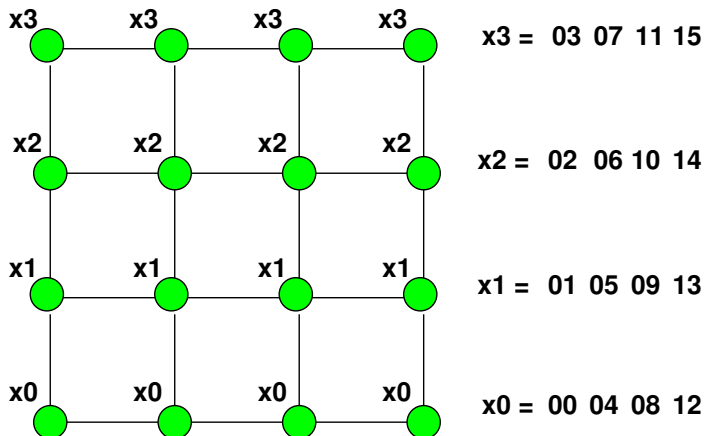
- Rozšíření algoritmu pro sítě na d dimenzí
- Po každé fázi (z celkového počtu d) je objem zpráv zdvojnásoben

ATA Redukce

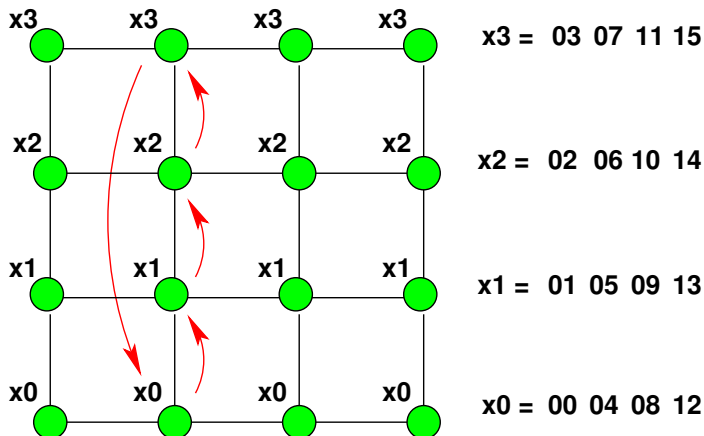
- Duální postup
- Po každé fázi je objem zpráv zredukován na polovinu

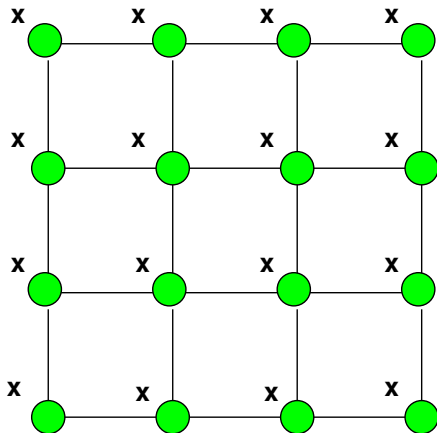




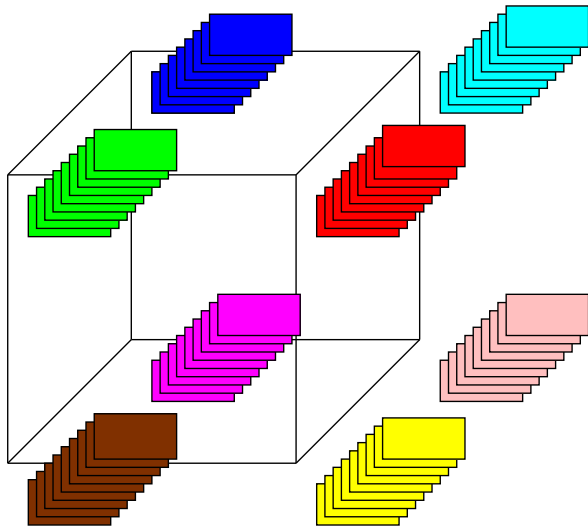


ATA Vysílání: Mřížka

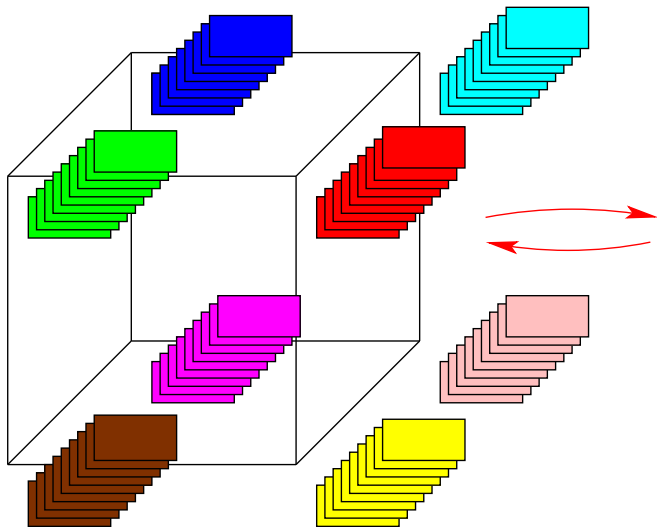


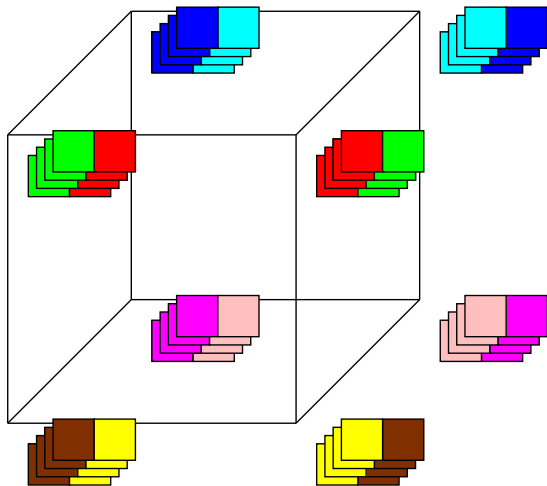


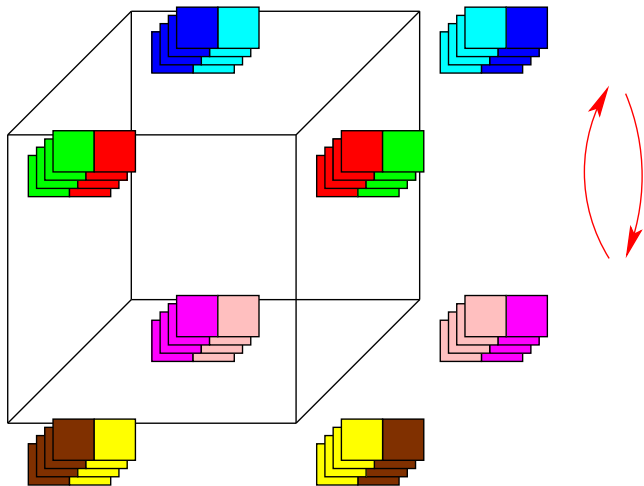
x = 03 07 11 15
02 06 10 14
01 05 09 13
00 04 08 12

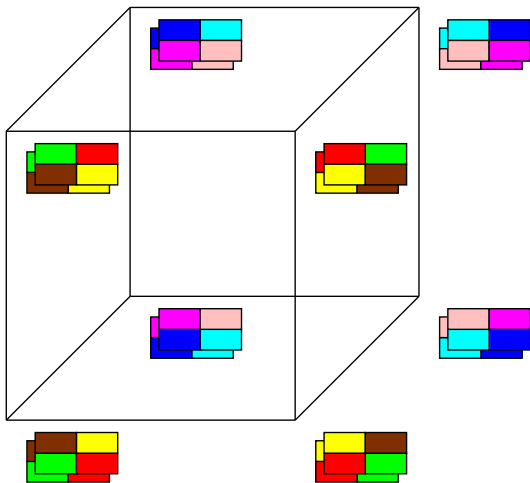


ATA Redukce: Hyperkostka

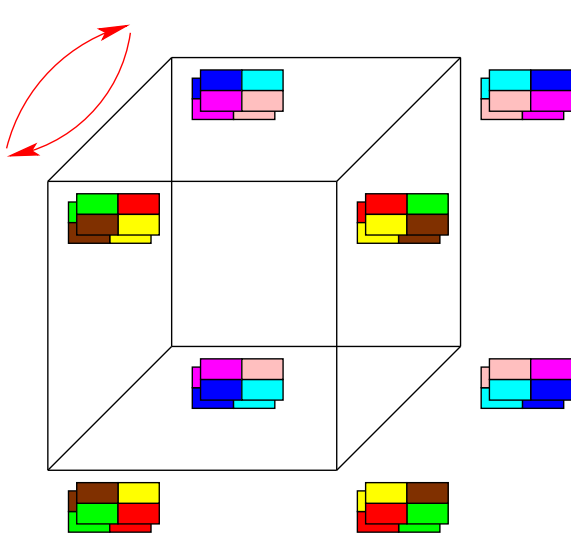


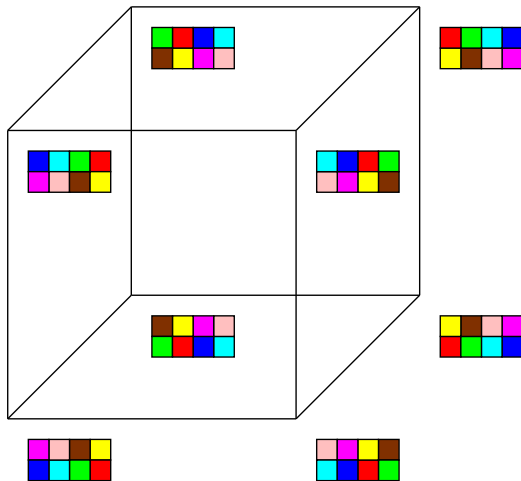






ATA Redukce: Hyperkostka





Kruh a lineární pole, p uzlů

- $T = (t_s + t_w m)(p - 1)$

2D mřížka, p uzlů

- 1. fáze $(t_s + mt_w)(\sqrt{p} - 1)$
- 2. fáze $(t_s + m\sqrt{p}t_w)(\sqrt{p} - 1)$
- Celkem: $T = 2t_s(\sqrt{p} - 1) + t_w m(p - 1)$

Hyperkostka, $p = 2^d$ uzlů

- $T = \sum_{i=1}^{\log p} (t_s + 2^{i-1} t_w m)$
- $T = t_s \log p + t_w m(p - 1)$

Pozorování

- Člen $t_w m(p - 1)$ se vyskytuje vždy
- Pipeline několika OTA operací

Všichni všem individuální komunikace

All-To-All individuální komunikace (ATA individuální)

- Každá úloha posílá různá data ostatním úlohám
- Dojde k výměně 2D pole zpráv ($p \times p$) v 1D prostoru (p)
- Také označováno jako "totální výměna"
- Změna struktury dat:

$$p * (m_1, \dots, m_p) \mapsto p * m_1, p * m_2, \dots, p * m_p$$

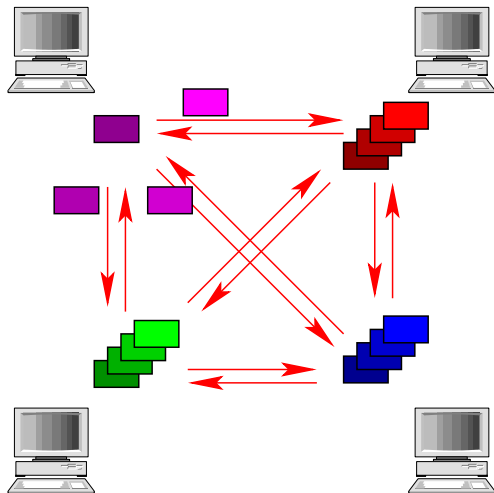
Příklad

- Transpozice matice ($A^T[j, i] = A[i, j]$)
- Matice $n \times n$ mapována po řádcích na n úloh
- Úloha j má k dispozici prvky $[j, 0], [j, 1], \dots [j, n - 1]$
- Úloha j chce znát prvky $[0, j], [1, j], \dots [n - 1, j]$

ATA Individuální: Změna struktury dat



ATA Individuální: Změna struktury dat



ATA Individuální: Změna struktury dat



Princip

- Nejprve všechny úlohy pošlou jedním směrem zprávy pro zbývajících $p - 1$ úloh
- V každém dalším kole každá úloha vyextrahuje zprávu, která je určena jí a zbývajících zprávy přepoše
- V posledním kole všechny úlohy přepošílají pouze jednu zprávu

Analýza ceny

- Počet kol je $p - 1$, všechny zprávy mají velikost m
- $$T = \sum_{i=1}^{p-1} (t_s + t_w m(p - i))$$
$$= t_s(p - 1) + \sum_{i=1}^{p-1} i t_w m$$
$$= (t_s + t_w m p / 2)(p - 1)$$

Princip

- Síť je \sqrt{p} řetízků o délce \sqrt{p}
- 1. fáze: mezi řetízky se vymění v jednom směru zprávy tak, aby informace pro každý uzel v řetízku byla někde v řetízku obsažena
- 2. fáze: v rámci řetízku se informace napropaguje na odpovídající místo

Příklad

- ATA individuální komunikace na síti 4x4 uzly

Cena

- Každá fáze distribuuje zprávy velikosti $m\sqrt{p}$ mezi \sqrt{p} uzly
- Cena jedné fáze (viz cena pro prsten): $(t_s + t_w mp/2)(\sqrt{p} - 1)$
- $T = (2t_s + t_w mp)(\sqrt{p} - 1)$

Princip

- Aplikace algoritmu pro d -dimenzionální síť
- Podél jedné dimenze se posílá vždy $p/2$ zpráv

Příklad

- Krychle $2 \times 2 \times 2$ uzlů

Cena

- V každé fázi vyměněno $mp/2$ dat
- $\log p$ fází

$$T = (t_s + t_w mp/2) \log p$$

- Navíc v každé fázi uzly přeuspořádávají/třídí zprávy

Problém

- Každý uzel posílá a přijímá $m(p - 1)$ dat
- Průměrná vzdálenost, na kterou data putují je $(\log p)/2$
- Celkový provoz na síti je $p \times m(p - 1) \times (\log p)/2$
- Počet linek v hyperkostce je $p(\log p)/2$
- Optimální algoritmus by měl dosáhnout složitosti

$$T = \frac{t_w pm(p - 1)(\log p)/2}{(p \log p)/2} = t_w m(p - 1)$$

Závěr

- Pro ATA individuální komunikaci není algoritmus pro síť ve tvaru mřížky optimální na sítích ve tvaru hyperkostky

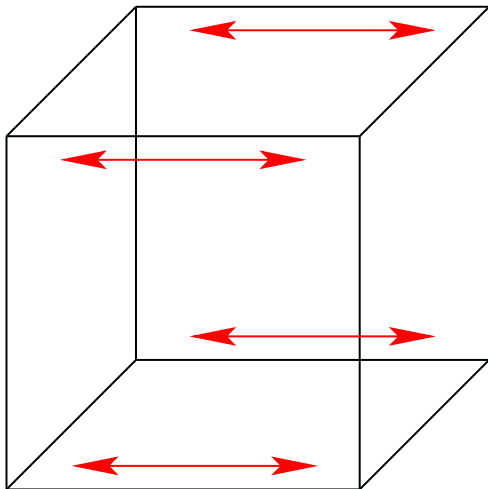
```
(1) proc ALL-TO-ALL-PERSONAL( $d, id$ )
(2)   for  $i := 1$  to  $2^d - 1$ 
(3)      $partner := id \text{ XOR } i$ 
(4)     send  $M_{id}$  to  $partner$ 
(5)     receive  $M_{partner}$  from  $partner$ 
(6)   end
(7) end
```

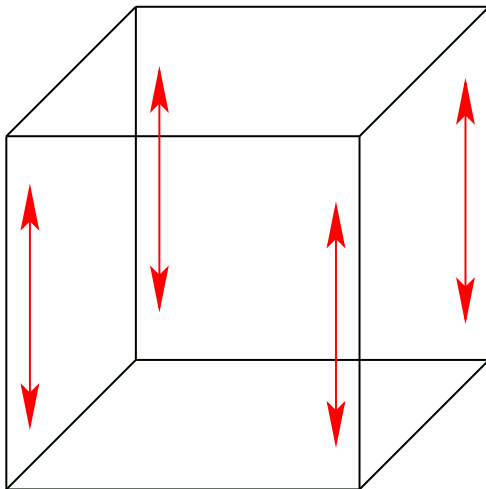
Pozorování:

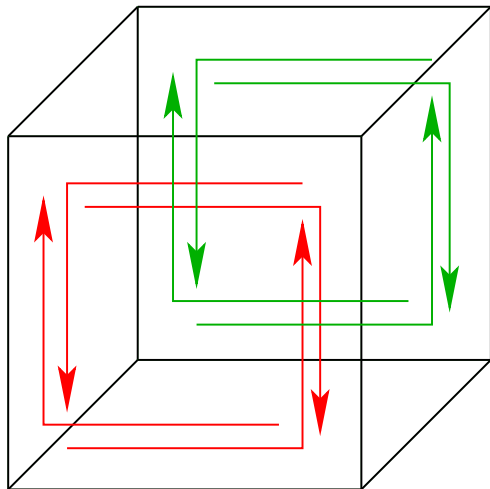
- Systematická a symetrická volba partnera
- Lze routovat tak, aby nedocházelo k blokování linek
- E-cube routing: Cesta mezi 2 body je dána pozicí odlišných bitů těchto bodů, routuje se od nejméně významného bitu
- Vyžaduje duplexní linky

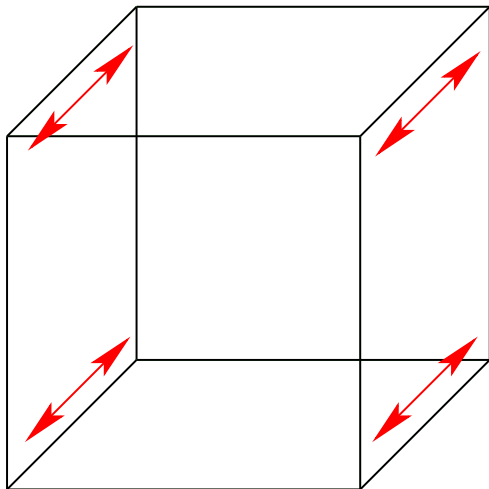
Cena:

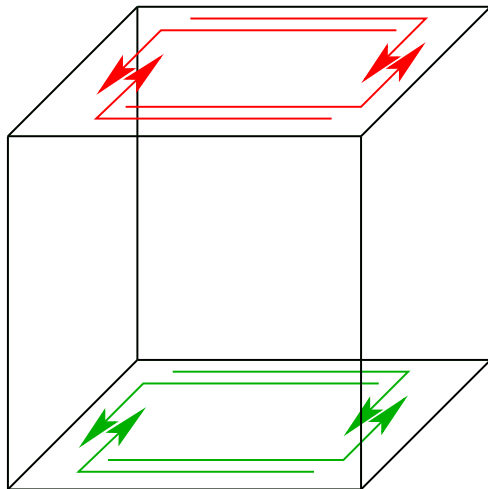
- $T = (t_s + t_w m)(p - 1)$

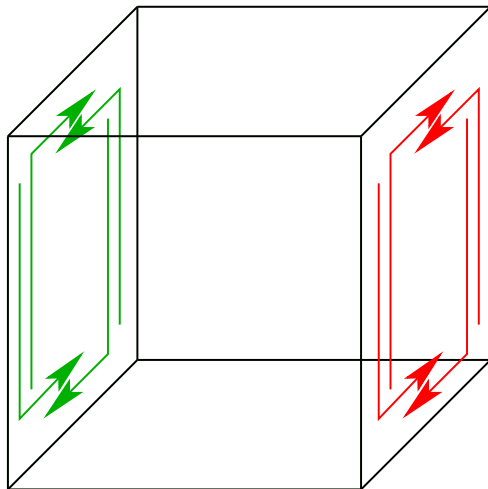












Operace kompletní redukce a prefixového součtu

Kompletní redukce (All reduce)

- Úlohy si vzájemně vyměňují data, která se kombinují
- Lze realizovat jako ATO redukci následovanou OTA vysíláním výsledku z předchozí redukce
- Změna struktury dat: $p * m \mapsto p * m$

Sémantika a využití pro účely synchronizace

- Úloha nemůže dokončit operaci, dokud všechny participující úlohy nepřispějí svým dílem
- Může být použito pro realizaci synchronizačního primitiva

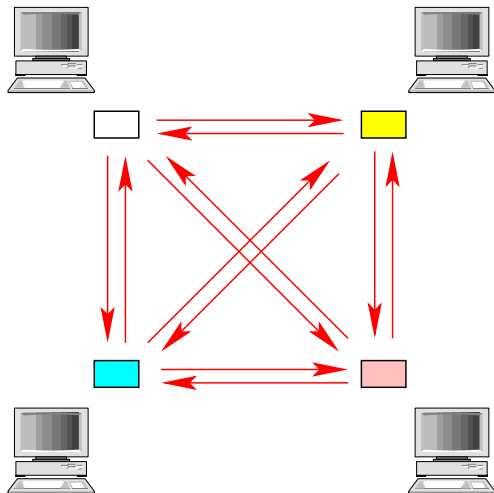
Implementace

- Naivně pomocí ATO a OTA, nebo
- Modifikací operace ATA vysílání
 - rozdíl od ATA: zprávy se nekumulují, ale kombinují
- Cena pro hyperkostku je $T = (t_s + t_w m) \log p$

Kompletní redukce: Změna struktury dat



Kompletní redukce: Změna struktury dat



Kompletní redukce: Změna struktury dat



Prefixový součet

- Úlohy posílají data ostatním úlohám se stejným či menším ID
- Data se kombinují (redukce)
- Změna struktury dat: $p * m \mapsto p * m$

Příklad

- Data v jednotlivých uzlech: $\langle 3, 1, 4, 0, 2 \rangle$
- Kombinace pomocí operace součtu
- Výsledná data $\langle 3, 4, 8, 8, 10 \rangle$

Implementace

- Použití algoritmu pro ATA
- Jak se pozná, že zpráva pochází od uzlu s menším ID?
- Všechny posílaná data obohacena o identifikátor původce

Scatter and Gather

Scatter (též označováno jako "Scan")

- Jedna úloha posílá unikátní zprávu každé další úloze
- One-To-All individuální (personalized) komunikace
- Na rozdíl od OTA vysílání se žádná data se neduplikují
- Změna struktury dat: $(m_1, \dots, m_p) \mapsto m_1, \dots, m_p$

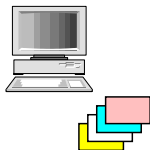
Gather (též označováno jako "Concatenation")

- Jedna úloha sbírá unikátní data od ostatních úloh
- All-To-One individuální (personalized) komunikace
- Na rozdíl od ATO redukce se nevyskytuje kombinace dat
- Změna struktury dat: $m_1, \dots, m_p \mapsto (m_1, \dots, m_p)$

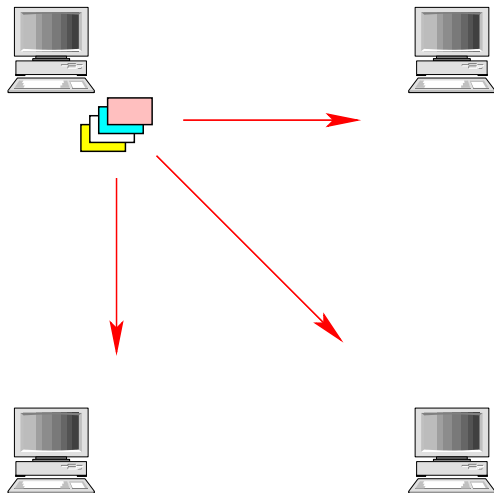
Implementace

- Využití algoritmů pro ATO a OTA
- Celkem $\log p$ fází, s každou fází se objem dat zvětšuje
- $T = t_s(\log p) + t_w m(p - 1)$

Scatter: Změna struktury dat



Scatter: Změna struktury dat



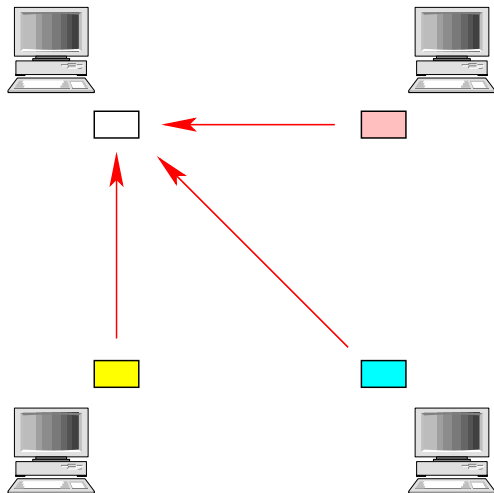
Scatter: Změna struktury dat



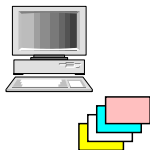
Gatter: Změna struktury dat



Gatter: Změna struktury dat



Gatter: Změna struktury dat



Cyklický posun

Permutace

- Obecnější komunikační primitiva
- Současně probíhající OTO přerozdělování dat
- Jedna úloha posílá data jedné jiné úloze

Příklad

- Cyklický posun o q (circular q -shift)
- p úloh
- Úloha i posílá data úloze $(i + q) \bmod p$

Použití

- Specifické maticové operace
- Vyhledávání vzorů v textu či obrazu

1-dimenzionální

- Intuitivně: rotace o q pozic
- Směr rotace závislý na q , určí se dle výrazu $\min(q, p - q)$

Dvourozměrné síť

- Akcelerace cyklického posunu s využitím druhé dimenze než, ve které probíhá posun
- Jedna dimenze má rozměr \sqrt{p} uzlů
- Posun v druhé dimenzi akceleruje o \sqrt{p} kroků

Cena

- Nejvzdálenější posun v jedné dimenzi je $\sqrt{p}/2$
- $T = (t_s + t_w m)(\sqrt{p})$

Příklad

- Síť 4x4 uzly, cyklický posun o 5

Princip

- Mapování lineárního pole na hyperkostku dimenze d
 - Zrcadlený šedý kód (reflected Grey code): $i = G(i, d)$
 - $G(0, 1) = 0$
 - $G(1, 1) = 1$
 - $G(i, x + 1) = \text{if } i < 2^x \text{ } G(i, x) \text{ else } 2^x + G(2^{x+1} - 1 - i, x)$
- q se vyjádří jako součet mocnin čísla 2
- Posun proběhne v tolika fázích, kolik je členů v součtu

Cena

- Uzly ve vzdálenosti mocniny čísla 2, jsou od sebe vzdáleny nejvýše 2 kroky (vlastnost kódu)
- Členů v součtu je nejvýše $\log p$
- Komunikace probíhá bez blokování linek
- $T = (t_s + mt_w)(2 \log p)$
- Se zpětným posunem je počet sčítanců max. $(\log p)/2$
- $T = (t_s + mt_w)(\log p)$

Binary Reflected Grey code

