Týden 3

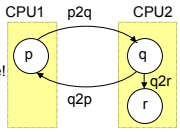
Programování zasíláním zpráv (MP)

- synchronní MP, projekt č.2
- programování v MPI

Synchronní komunikace (CSP,Transim)

```
CHAN OF ANY p2q,q2p,q2r: -- deklarace kanálů lib. typ dat
[16]CHAN OF ANY ring: -- pole 16 kanálů, int. i externí dohromady
```

```
PLACED PAR
  INT e,...:          -- lokální proměnné v p
  SEQ | p
    p2q ! e | m       -- send e (INT),
    ...               -- simuluje přenos m byte!
  INT v,...:          -- lokální proměnné v q
  SEQ | q
    p2q ? v           -- receive into v
    ...
```



ext. kanály : p2q, q2p
int. kanály : q2r

```
MAP -- mapping sw to hw
  MAP CPU1: p
  MAP CPU2: q,r
```

Nesprávný kód (Transim)

```
INT x:
PAR
  SEQ | p
    x:=y-1
  SEQ | q
    y:=x+2
  SEQ | r
    ch ? x
```

```
PAR
  INT v:                INT r,s:
  SEQ | alfa            SEQ | beta
    a2b ! 42              b2a ! r
    b2a ? v               a2b ? s
```

deadlock!

Sdílené proměnné v komponentách PAR: proměnná, do níž se přiřazuje nebo která je modifikována vstupem se může vyskytovat jen v jedné komponentě PAR, jelikož pořadí zpracování není známo.

Deadlock: záleží na pořadí komunikací. Použij ? || ! v jednom nebo obou procesech alfa, beta.

Výběr kanálu (Ada, CSP, Transim)

Jelikož synchronní receive čeká na partnera, seznam vstupů se nedá prohlížet sekvenčně. Nový příkaz ALT připomíná CASE se strážemi:

```
ALT
  ch1 ? x          -- stráž: vstup, průchozí když partner rdy
    ... task1      -- strážený příkaz
  (y<0)&ch2 ? x    -- stráž: podmínka & receive
    ... task2
  (y>0)&SKIP       -- stráž: pouze podmínka, bez receive
    ... task3
  clock ? AFTER t-- průchozí při údaji časovače >= t
    ... task4
  TRUE&SKIP        -- stráž vždy průchozí
    ... task5      -- jen na konci PRI ALT!
```

Implementace ALT

1. jsou vyhodnoceny logické podmínky ve strážích;
2. je zjištěna připravenost vstupních kanálů; proces je pozastaven (do stavu S-ALT) až do doby kdy je průchozí alespoň jedna stráž;
3. vybere se (náhodně) jedna povolená alternativa a provede se, ostatní alternativy se již neuvažují;

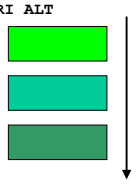
Omezení:

- stejný kanál se nemůže vyskytovat ve dvou strážích;
- výstup (send) se ve strážích použít nemůže : ALT(!) a ALT(?) by spolu nemohly komunikovat vlivem nedeterminismu na obou stranách;
- ALT nemůže být ve stráženém příkazu (ALT nelze vnořovat).

Priorita u PAR a ALT

```
PRI PAR
  SEQ | high
  SEQ | low
```

```
PRI ALT
```

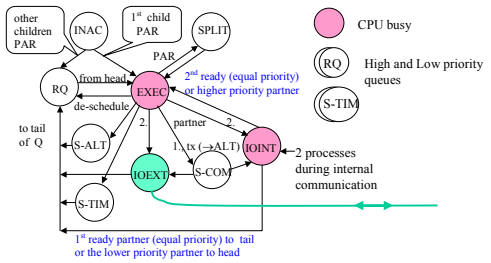


high: až do konce, pokud nenarazí na komunikaci

low: běží po časové kvantum

provede se první připravená alternativa v textovém pořadí

Stavový diagram procesů (Transim)

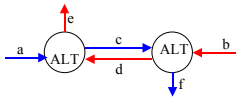


Deadlock u příkazů ALT

(Příklady: deadalt.in, dead-ok.in)

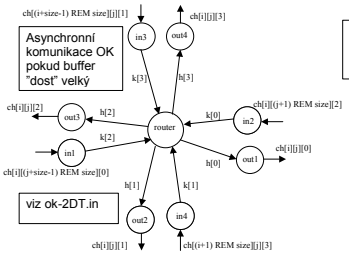
```
WHILE TRUE                    WHILE TRUE   -- "deadalt.in"
  ALT                           ALT
    a ? x                         b ? y
      c ! x                         d ! y
    d ? x                         c ? y
      e ! x                         f ! y
```

Je-li v obou příkazech ALT vybrána první alternativa, vznikne deadlock, protože partner pro komunikaci je ve druhé alternativě !



Náprava:
proces buffer pro každý kanál a, b, c, d! → (asynchr. komunikace)
"dead-ok.in"

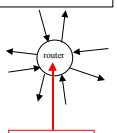
Síť kanálů na topologii 2D-torus



Asynchronní komunikace OK pokud buffer "dost" velký

viz ok-2DT.in

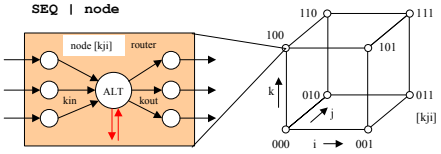
Synchr. komunikace může uvázvnout, viz bad-2DT.in



Nebezpečí deadlocku u ALT

Síť kanálů na hyperkostce

```
[2][2][2][3]CHAN OF ANY ch: -- outgoing channels
PLACED PAR i=0 FOR 2        -- ch[i][j][k][*]
  PLACED PAR j=0 FOR 2
    PLACED PAR k=0 FOR 2    -- myid:= 4*k+2*j+i
      SEQ | node
```




```
ch[(i+1)REM 2][j][k][0]
ch[i][(j+1)REM 2][k][1]
ch[i][j][(k+1)REM 2][2]
```

```
ch[i][j][k][0]
ch[i][j][k][1]
ch[i][j][k][2]
```

Klasifikace komunikačních úloh

Komunikace Jeden s jedním (p2p, point-to-point)

- komunikace jednoho páru (nesousedních uzlů) - bez deadlocku!
- permutační směrování: každý uzel i posílá jednu zprávu uzlu j, zobrazení i → j je permutace
- násobné směrování: podmnožina permutačního směrování

Komunikace Jeden (One) s mnoha nebo všemi (All)

- rozhlášení (Broadcast) jeden všem, OAB: stejná zpráva všem uzlům
- multicast, MC: rozhlášení do libovolné podmnožiny uzlů
- rozptýlení (Scatter) jeden všem, OAS: jiná zpráva každému uzlu

Klasifikace komunikačních úloh - pokrač.

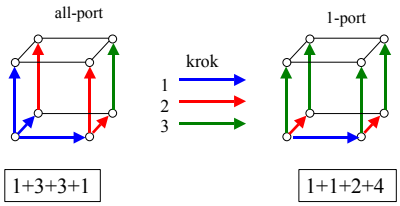
Komunikace všichni s jedním

- posbírání (Gather) jedním ode všech, AOG: všechny uzly pošlou různé zprávy jednomu uzlu (root). Je-li spojeno s kombinací výsledků: AO Reduce
- AOG může být iniciováno rozhlášením zprávy OAB z root - všem

Komunikace všichni se všemi

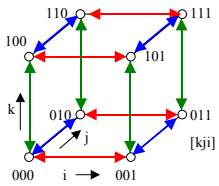
- rozhlášení všichni všem, AAB : všechny uzly provádějí současně OAB (gossip, drby)
- rozptýlení - posbírání, všichni ode všech, AAS (AAG): všechny uzly provádějí současně OAS (AOG) (total exchange). Spojeno s kombinováním zpráv (výsledků): AA Reduce

Rozhlášení OAB na SF hyperkostce



$T_{OAB} = (t_s + m\, t_w) \log P$

Trvání komunikace AAB na hyperkostce



P = 8, 3 kroky, zprávy m slov

1	$t_s + 1\, m\, t_w$
2	$t_s + 2\, m\, t_w$
3	$t_s + 4\, m\, t_w$

$1+2+4+\ldots+P/2 = P-1 \quad (P = 2^k)$

$T_{AAB} = t_s \log P + m\, t_w (P-1)$

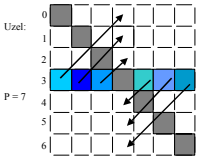
Předpoklady: - full duplex, 1 brána, *kombinující* model
- 1 brána stačí (všechny brány nic nezrychlí)
- SF nebo WH (není rozdíl, pouze mezi sousedy)

AAB s paralelní redukcí: viz "add-cube.in".

AAS: všichni všem různé zprávy

Nekombinující uzly :

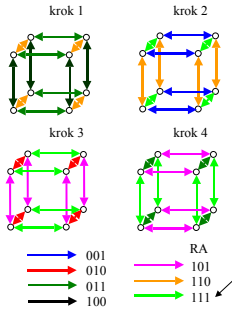
- každý uzel posílá P-1 různých zpráv, každá m slov
- celkový počet zpráv = P(P-1)
- analogie: transpozice matice A[P×P], rozdělené po řádcích na procesory; uzel i zašle prvek a_{ij} uzlu j (prvek a_{ii} sám sobě)



Komu odeslat:
id adresáta =
sloupec odesilatele

Kam uložit:
sloupec u adresáta =
id (řádek) odesilatele

Nekombinující AAS na SF hyperkostce



Plán přesunů:

	dimenze		
krok	0	1	2
1	3	6	4
2	1	7	6
3	7	2	5
4	5	3	7

Relativní adresa zprávy
RA = src ⊕ dst

$T_{AAS} = (P/2)(t_s + m t_w + t_d)$

Message Passing Interface Standard (MPI)

Není jazyk, ale specifikace knihovních programů volaných z programu.

Výhoda: *přenositelnost programů* (běží na svazcích pracovních stanic, síti PC, SMP, jednom PC, prakticky na všech paralelních systémech).

MPI 1.1 (1993-94) (pro Fortran 77 nebo C)

- souběžné komunikující statické procesy na definovaných procesorech
- skupiny procesů, virtuální topologie
- nový pojem *komunikátoru*
- kolektivní komunikace a agregace

MPI 2.0: 1995-97 (pro Fortran 90 nebo C++)

- dynamické procesy
- jednostranná komunikace (remote memory access, RMA)
- paralelní I/O
- podpora zpracování stylem klient - server

MPI - odkazy

MPI Forum: sdružení výrobců, uživatelů a programátorů.
MPI Home page:
http://www-unix.mcs.anl.gov/mpi/

ONLINE COURSE and course material to MPI,
OpenMP, HPF, and advanced topics in parallel programming.
http://www.hlrs.de/organization/par/par_prog_ws/
- některé části s anglickým zvukovým doprovodem

Procesy a skupiny procesů v MPI

- MPI 1.1 neříká nic o tom, jak se na začátku vytvoří procesy, to je ponecháno na implementaci. Předpokládají se statické procesy, všechny se vytvoří v okamžiku načtení paralelního programu a existují až do ukončení celého programu. Implicitní skupina všech takových procesů je

  MPI_COMM_WORLD - all MPI tasks (default)

  Jejich počet je parametr při načítání programu. Žádný proces nemůže být vytvořen nebo ukončen během provádění programu.
- aplikace MPI = n procesů (tasks), na $1 \leq P \leq n$ procesorech; každý má svůj index, tzv. *rank* (INT od 0 do n-1), a to v kontextu určité skupiny procesů; některý proces může být ve více skupinách.
- Pojmenovaná *skupina procesů* (*task group*) = uspořádaná množina indexů; deklarace: MPI_Group taskforce; /*popisovac skupiny*/

Skupiny procesů a komunikátory v MPI

- Komunikátor = skupina procesů plus kontext; definují se procesy, které spolu v určité fázi programu komunikují. Deklarace:

  MPI_Comm MyWorld, SplitWorld; /*popisovac komunikatoru*/
- Předdefinované komunikátory se nemusí deklarovat:

  MPI_COMM_WORLD

  MPI_COMM_SELF
- Nový komunikátor MyWorld se stejnou skupinou procesů jako MPI_COMM_WORLD se vytvoří voláním :

  MPI_Comm_dup (MPI_COMM_WORLD, &MyWorld)
- Proces může zjistit svůj index ve skupině a počet procesů ve skupině voláním

  MPI_Comm_size (MyWorld, &group_size);

  MPI_Comm_rank (Myworld, &my_rank);
- Skupina procesů taskforce v komunikátoru MyWorld (bez kontextu):

  MPI_Comm_group (MyWorld, &taskforce)

Proč potřebujeme komunikátory?

Komunikace je složitější než jen

send (&s, Q); → **recv(&r, P);**

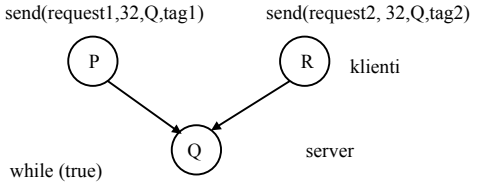
Zprávy odeslané za sebou mohou přijít do cíle v opačném pořadí. Aby nedošlo k nesprávné interpretaci, je potřeba zprávu rozšířit:

- obsah zprávy, buffer, je určen adresou a délkou zprávy
- obálka zprávy = destinace + **příznak** (tag)

send (&s, 16, Q, tag);

Příznaky umožňují selekci zpráv přicházejících na jednu bránu v různých časech (ALT byla selekce prostorová, ze zpráv na několika vstupech).

Použití příznaků při zasílání zpráv




```
while (true)
recv(received_request, Any_Process, 32, Any_Tag, Status);
if (Status.Tag==tag1) zpracuj zadost jednim zpusobem;
if (Status.Tag==tag2) zpracuj zadost druhym zpusobem;
```