technische universität
dortmund

fakultät für informatik
informatik 12

Synthese Eingebetteter Systeme

Sommersemester 2011

Übung 5

Michael Engel
Informatik 12
TU Dortmund

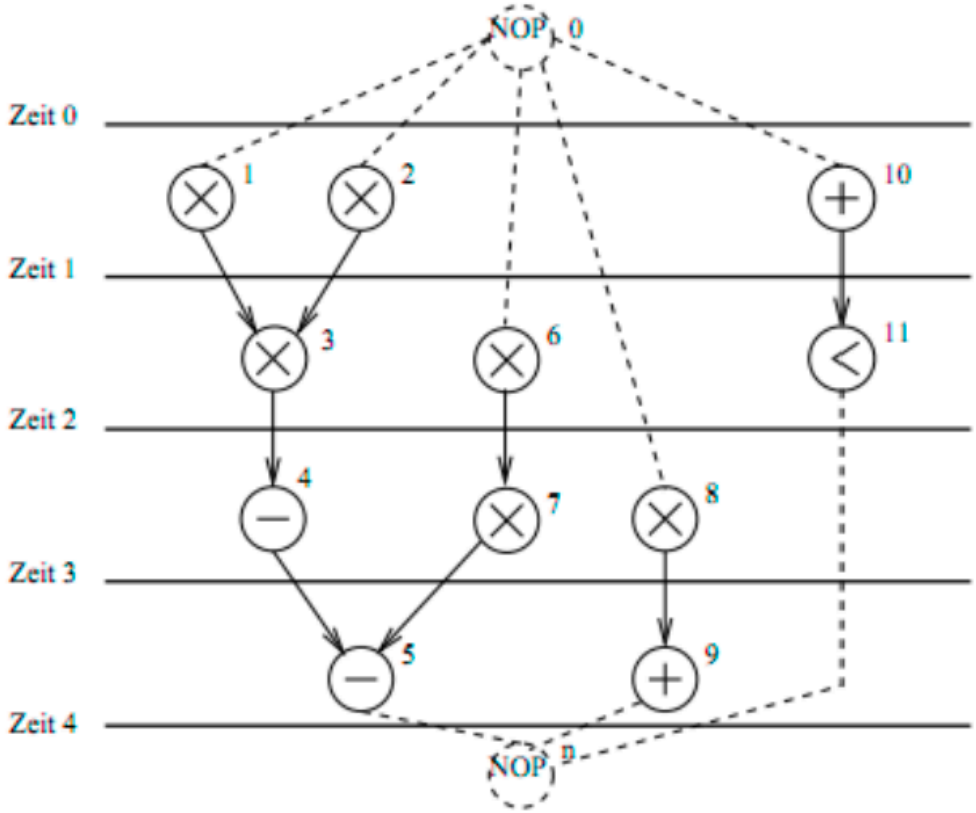
2011/07/01

Übung 5

- Bereitstellung
- Amdahl's Law
- Abbildung auf Architekturen mit verteilten Speicher

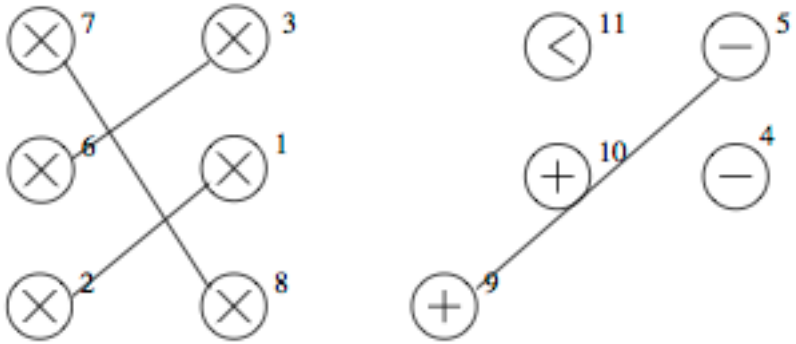
Bereitstellung

- Gegeben: DFG
- Es existieren zwei Ressourcentypen:
- Multiplizierer
- ALU mit den Operationen
  - Addition (+)
  - Subtraktion (–)
  - Vergleich (<)

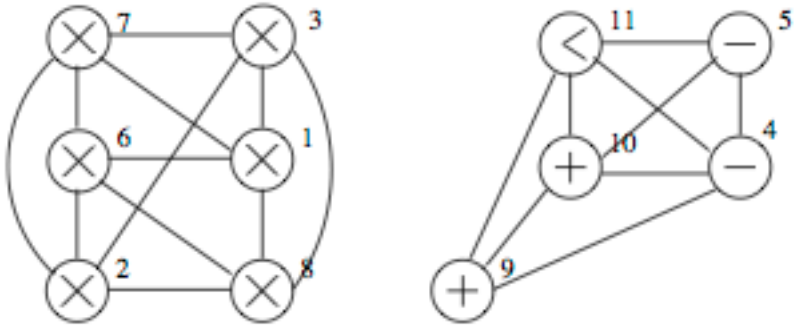


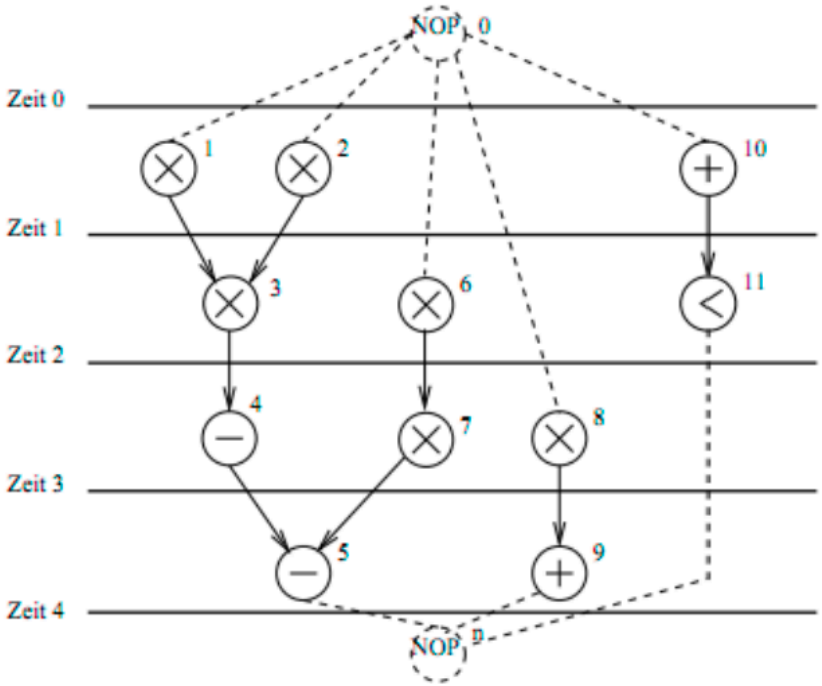
Bereitstellung

- Erstellen Sie den Konfliktgraphen



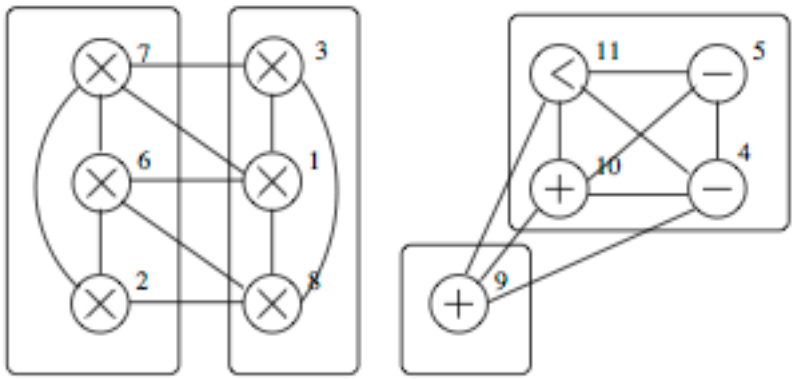
- und den Kompatibilitätsgraphen



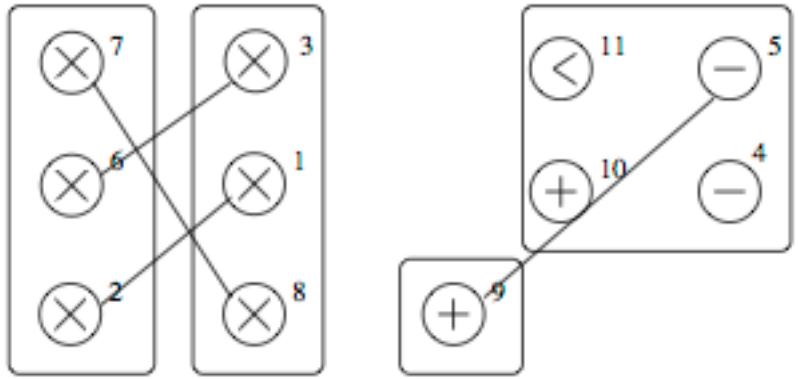


Bereitstellung

- Bestimmen Sie eine Cliquenpartition des Kompatibilitätsgraphen

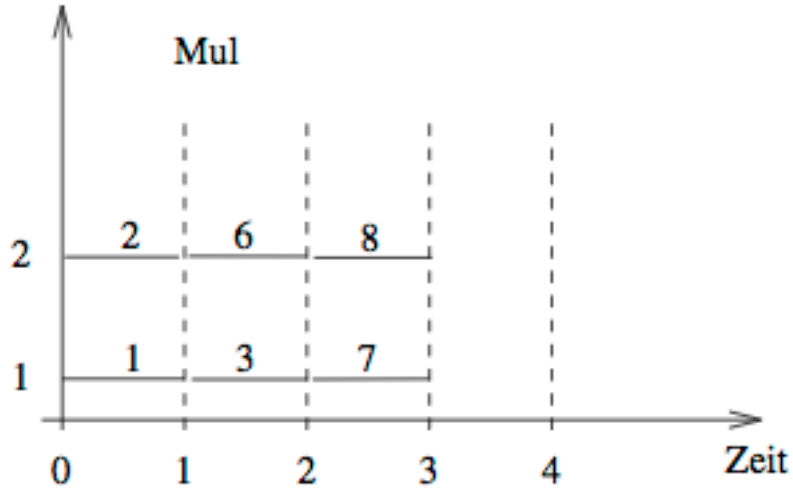


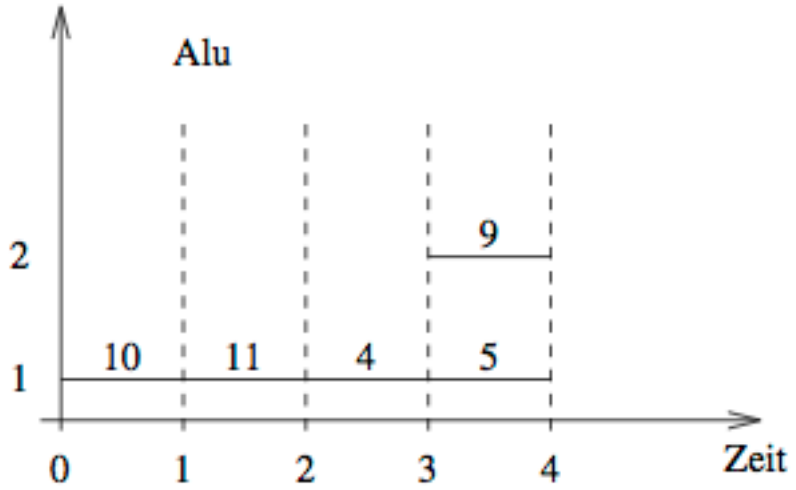
- und eine Färbung des Konfliktgraphen



Bereitstellung

- Berechnen Sie eine optimale Färbung des Konfliktgraphen mit dem Leftedge-Algorithmus





Bereitstellung

- Wie viele Ressourcen werden von jedem Typ benötigt?
- Man muss die minimale Anzahl von Cliquen bzw. minimale Anzahl von Farben des Konfliktgraphen bestimmen.
- In dieser Aufgabe sind dies
  - 4 Cliquen bzw. 4 Farben für den Ressourcentyp r1
  - 3 Cliquen (3 Farben) für den Ressourcetyp r2.

Amdahl's Law

- Jedes Quadrat eine unteilbare Aufgabe
- Grau hinterlegte Quadrate: nicht parallisierbare Aufgaben
- Die anderen sind parallelisierbar
- Eine solche Aufgabe benötigt auf einem Prozessor exakt eine Zeiteinheit

Amdahl's Law

- Bestimmen Sie mit Hilfe von Amdahl's Law:
  - 1. den parallelisierbaren Anteil des Programms

  - g = 15/20 = 75%

Amdahl's Law

- Bestimmen Sie mit Hilfe von Amdahl's Law:
  - 1. den maximalen Speedup für die Prozessoranzahlen p = ∞; p = 5; p = 4; p = 3 und p = 2

  - p = ∞: Smax= 1/f = 20/5 = 4
  - p = 5: Smax = 1/(0,25+0,75/5) = 2,5
  - p = 4: Smax = 1/(0,25+0,75/4) ≈ 2,29
  - p = 3: Smax = 1/(0,25+0,75/3) = 2
  - p = 2: Smax = 1/(0,25+0,75/2) = 1,6

Amdahl's Law

- Kann der in 2. berechnete Speedup tatsächlich erreicht werden?

- Dieser Speedup kann nicht erreicht werden, da die parallelisierbaren Teile zum einen nicht teilbar und zum anderen nicht zusammenhängend sind
- Damit können die Teile nicht optimal auf die p Prozessoren verteilt werden
- Amdahl's Gesetz stellt lediglich eine obere Schranke dar!
- Im Spezialfall p = ∞ wird angenommen, dass der parallelisierbare Anteil $\rightarrow 0$ reduziert wird
  - unendlich viele Prozessoren erledigen die in unendlich kleine Teile zerlegten Tasks in unendlich kleiner Zeit

Abbildung auf Architekturen mit verteilten Speicher

- Berechnen Sie die Determinante einer 3x3-Matrix auf einer Architektur mit verteiltem Speicher

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} \Rightarrow \begin{vmatrix} x_{0,0} & x_{0,1} & x_{0,2} \\ x_{1,0} & x_{1,1} & x_{1,2} \\ x_{2,0} & x_{2,1} & x_{2,2} \end{vmatrix}$$

$$\det = a * (e*i - f*h) + b * (f*g - d*i) + c * (d*h - e*g)$$

$$\begin{aligned} \det = {} & x[0,0] * (x[1,1]*x[2,2] - x[1,2]*x[2,1]) \\ & + x[0,1] * (x[1,2]*x[2,0] - x[1,0]*x[2,2]) \\ & + x[0,2] * (x[1,0]*x[2,1] - x[1,1]*x[2,0]) \end{aligned}$$

Abbildung auf Architekturen mit verteilten Speicher

det = x[0,0] * (x[1,1]*x[2,2] – x[1,2]*x[2,1])

+ x[0,1] * (x[1,2]*x[2,1] – x[1,0]*x[2,2])

+ x[0,2] * (x[1,0]*x[2,1] – x[1,1]*x[2,0])

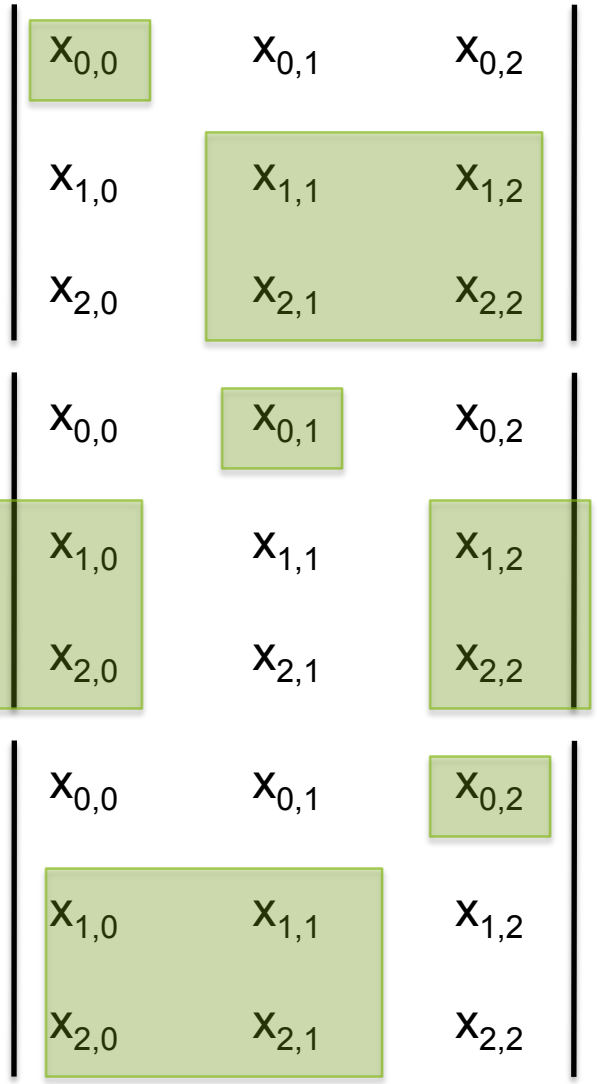
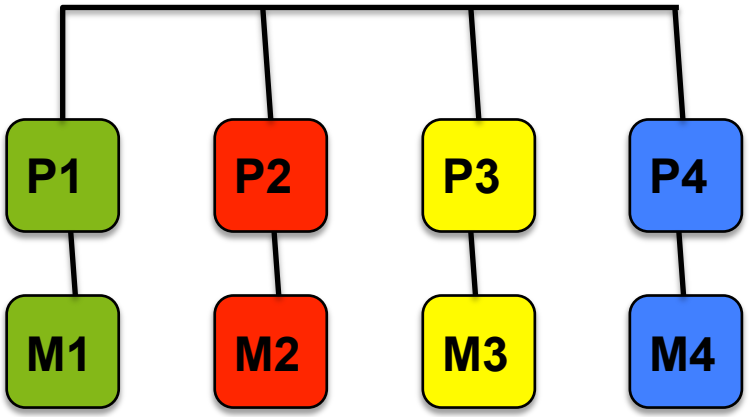


Abbildung auf Architekturen mit verteilten Speicher

- System mit 4 Prozessoren:
- “Master” (P1) verteilt Aufgaben, sammelt Teilergebnisse, berechnet Endergebnis
- “Worker” (P2-P4) berechnen Determinanten der 2x2-Teilmatrizen:



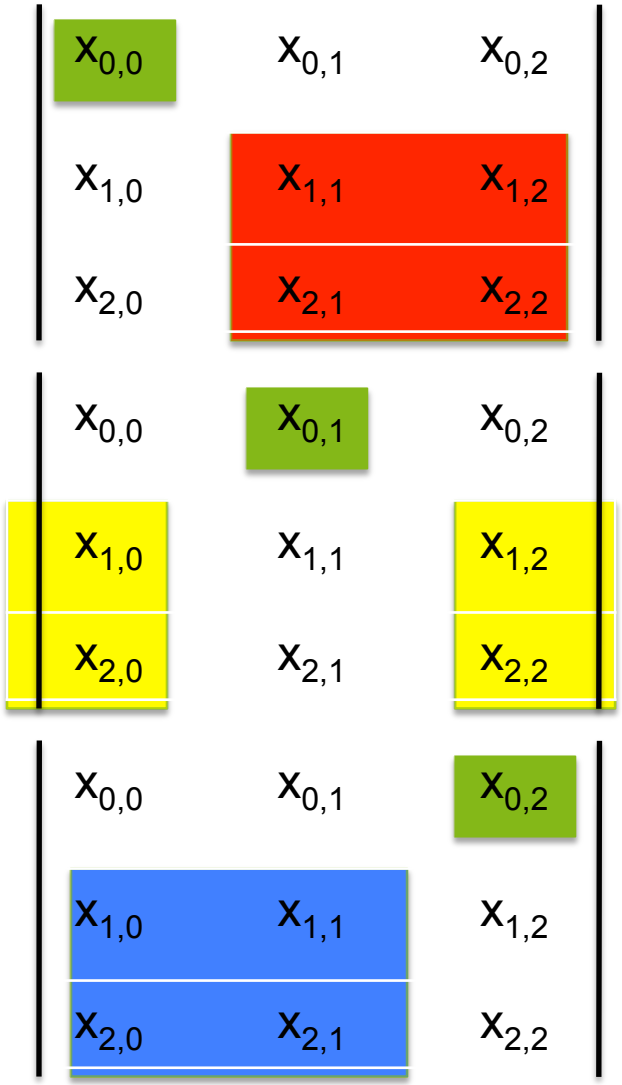
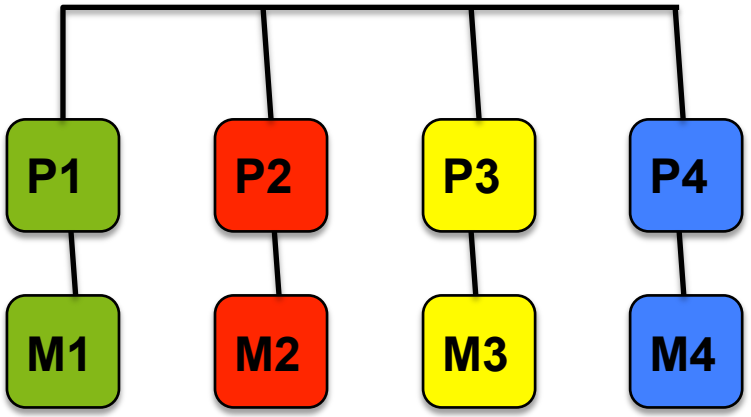


Abbildung auf Architekturen mit verteilten Speicher

- Algorithmus (Pseudocode)




```
Array A[3,3]
Float Det[3]

for (x = 0 to 2)  -- Spalte
   Switch(x)
     Case 0: i1=1; i2=2;
     Case 1: i1=0; i2=2;
     Case 2: i1=0; i2=1;
   Det[x] = A[0,i1]*A[1,i2] – A[0,i2]*A[1,i1]

Det = x[0,0]*Det[0] + x[0,1]*Det[1] + x[0,2]*Det[2]
```

Abbildung auf Architekturen mit verteilten Speicher

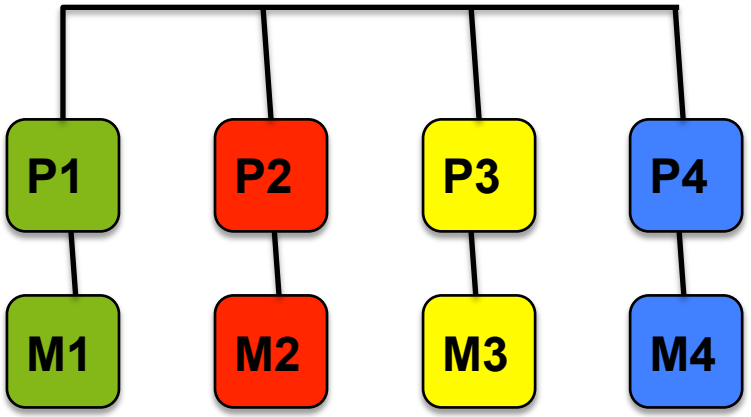
- Master

```
Array A[3,3]
Float Det[3]

for (x = 0 to 2)  -- Spalte
   Switch(x)
     Case 0: i1=1; i2=2;
     Case 1: i1=0; i2=2;
     Case 2: i1=0; i2=1;
   Send(x+1, A[0,i1], A[1,i2], A[0,i2], A[1,i1])

for (x = 0 to 2)  -- Spalte
   Receive(Det[x])

Det = x[0,0]*Det[0]
    + x[0,1]*Det[1]
    + x[0,2]*Det[2]
```



- Worker

```
Float A,B,C,D,Det

Receive(A,B,C,D)
Det = A*B-C*D
Send(0, Det)
```

Abbildung auf Architekturen mit verteilten Speicher

- Master: Alternative

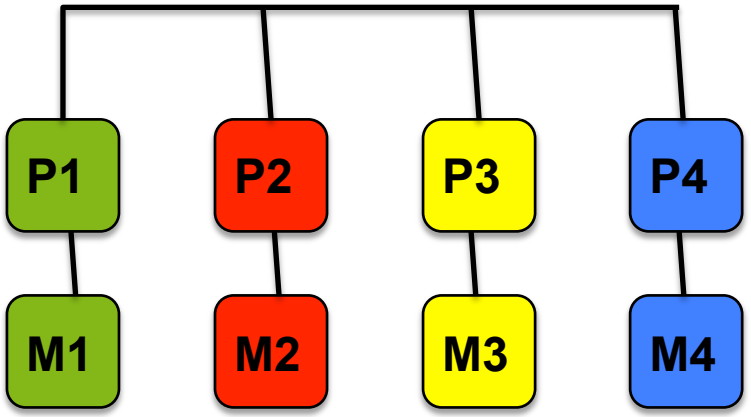
```
Array A[3,3]
Float Det[3]

for (x = 0 to 2)  -- Spalte
   i1 = (x==0)?1:0;
   i2 = (x==2)?1:2;

   Send(x+1, A[0,i1], A[1,i2], A[0,i2], A[1,i1])

for (x = 0 to 2)  -- Spalte
   Receive(x+1, Det[x])

Det = x[0,0]*Det[0]
    + x[0,1]*Det[1]
    + x[0,2]*Det[2]
```



- Worker

```
Float A,B,C,D,Det

Receive(A,B,C,D)
Det = A*B-C*D
Send(0, Det)
```

Fragen?

- *Fragen!*