Data Parallel Programming II

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC-Linz)
Johannes Kepler University, A-4040 Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine

SIMD Mesh Matrix Multiplication

Algorithm: see slide

- n^2 processors,
- $3n$ time.

SIMD Mesh Matrix Multiplication

Maspar program: see slide

1. Precondition array
   - Shift row i by $i-1$ elements west,
   - Shift column j by $j-1$ elements north.
2. Multiply and add

   On processor $\langle i, j\rangle$:

$$c = \sum_k a_{ik} * b_{kj}$$

- Inverted dimensions
  - Matrix $\downarrow i, \rightarrow j$.
  - Processor array $\downarrow$ `iyproc`, $\rightarrow$ `ixproc`.
- $4n$ shift and n arithmetic operations.
- n^2 processors.

SIMD Mesh Matrix Multiplication

Problem

- Processor array of fixed size $(2^d \times 2^d)$,
- Matrix size arbitrary $(2^n \times 2^n, n > d)$.

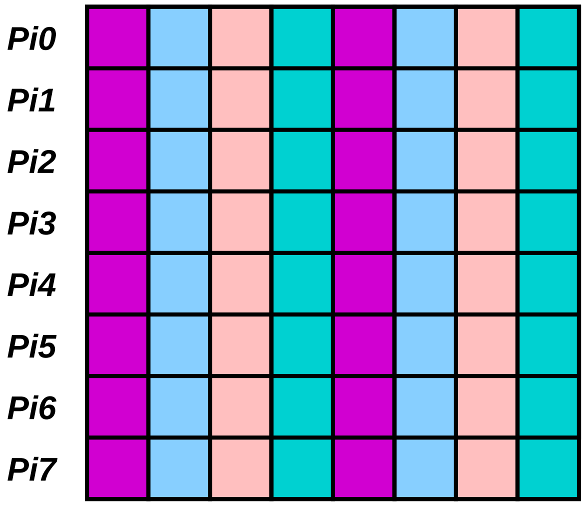
Solution

Circular mapping of matrix columns (A) resp. rows (B) to processors.

Processor Array

A_{00} A_{04}	A_{01} A_{05}	A_{02} A_{06}	A_{03} A_{07}
A_{10} A_{14}	A_{11} A_{15}	A_{12} A_{16}	A_{13} A_{17}
A_{20} A_{24}	A_{21} A_{25}	A_{22} A_{26}	A_{23} A_{27}
A_{30} A_{34}	A_{31} A_{35}	A_{32} A_{36}	A_{33} A_{37}

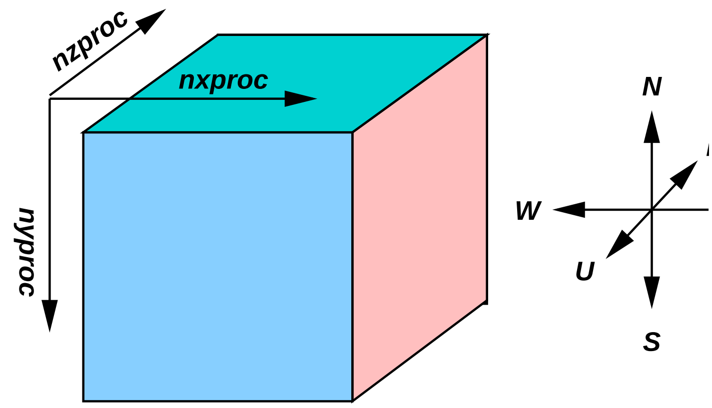
Matrix A



Analogous for matrix B.

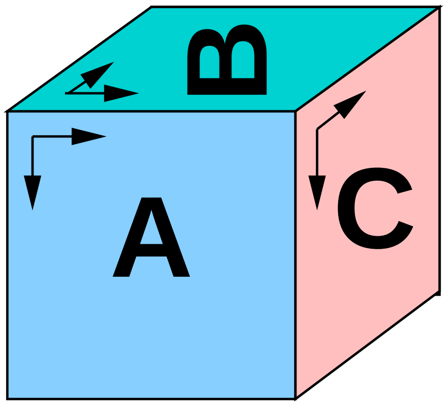
SIMD Cube Matrix Multiplication

Cube of d^3 processors



Idea

- Map $A(i,j)$ to all $P(j,i,k)$
- Map $B(i,j)$ to all $P(i,k,j)$



SIMD Cube Matrix Multiplication

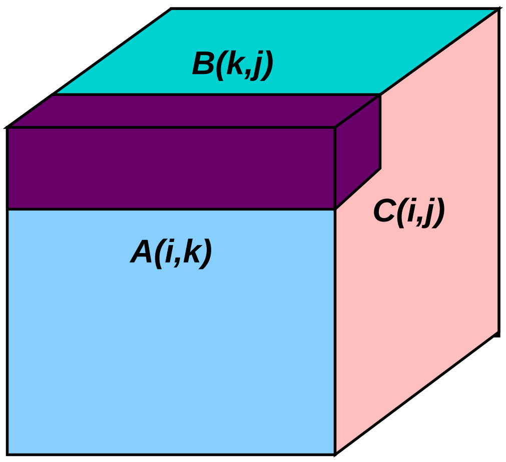
Multiplication and Addition

- Each processor computes single product

  $P_{ijk} : c_{ijk} = a_{ik} * b_{k,j}$

- Bars along x-directions are added

  $P_{0ij} : C_{ij} = \Sigma_k c_{ijk}$



SIMD Cube Matrix Multiplication

Maspar Program

```
int A[N,N], B[N,N], C[N,N];
plural int a, b, c;

a = A[iyproc, ixproc];
b = B[ixproc, izproc];
c = a*b;

for (i = 0; i < N-1; i++)
  if (ixproc > 0)
    c = xnetE[1].c
  else
    c += xnetE[1].c;

if (ixproc == 0) C[iyproc, izproc] = c;
```

- $O(n^3)$ processors,
- $O(n)$ time.

SIMD Cube Matrix Multiplication

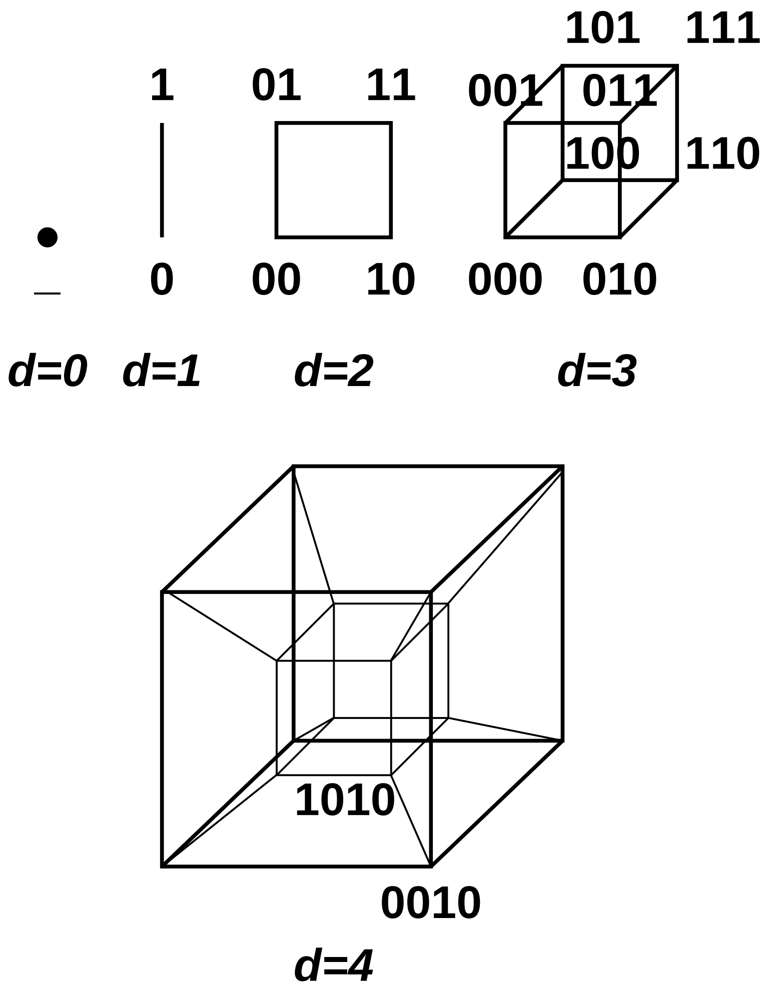
Tree-like summation

```
plural x, d;
...

x = ixproc;
d = 1;
while (d < N) {
  if (x % 2 == 0)
    c += xnetE[d].c;
  else
    x /= 2;
  d *= 2;
}

if (ixproc == 0) C[iyproc, izproc] = c;
```

- $O(\log n)$ time
- $O(n^3)$ processors

Long-distance communication required!

SIMD Hypercube Mat. Multiplication



- d-dimensional hypercube $\Rightarrow$ processors indexed with d bits.
- p_1 and p_2 differ in i bits $\Rightarrow$ shortest path between p_1 and p_2 has length i.

SIMD Hypercube Matrix Multiplication

Mapping of cube with dimension n to hypercube with dimension d.

- Hypercube of $n^3 = 2^d$ processors $\Rightarrow d = 3s$ (for some s).
- 64 processors $\Rightarrow n = 4, d = 8, s = 3$.

Hypercube	$\underline{d_5 d_4}$	$\underline{d_3 d_2}$	$\underline{d_1 d_0}$
Cube	x	y	z

- Embedding algorithm
  - Cube indices in binary form (s bits each)
  - Concatenate indices ($3s = d$ bits)
- Neighbor processors in cube remain neighbors in hypercube.
- Any cube algorithm can be executed with same efficiency on hypercube.

SIMD Hypercube Matrix Multiplication

Tree summation in hypercube.

Processors	000	001	010	011	100	101	110	111
Step 1	r_0	s_0	r_1	s_1	r_2	s_2	r_3	s_3
Step 2	r_0		s_0		r_1		s_1	
Step 3	r_0				s_0			

- Each processor receives value from neighboring processors only.
- Only short-distance communication is required.

Cube algorithm can be more efficient on hypercube!