

CS575 Parallel Processing

Lecture six: Dense Matrix Algorithms
Linear equations
Wim Bohm, Colorado State University

Except as otherwise noted, the content of this presentation is licensed under the Creative Commons Attribution 2.5 license.

Mapping $n \times n$ matrix to p PEs

- Striped: allocate rows (or columns) on PEs
 - Block striped: consecutive rows to one PE, e.g.:
PE# 0 0 0 0 1 1 1 1 2 2 2 2 3 3 3 3 3
Row 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
 - Cyclic striped: interleaving rows onto PEs
PE# 0 1 2 3 0 1 2 3 0 1 2 3 0 1 2 3
Row 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
 - Hybrid
PE# 0 0 1 1 2 2 3 3 0 0 1 1 2 2 3 3
Row 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
 - Finest granularity
 - One row (or column) per PE, ($p = n$)

CS575 lecture 7

2

Mapping $n \times n$ matrix to p PEs (cont.)

- Blocked / Checkerboard
 - Map $n/\sqrt{p} \times n/\sqrt{p}$ blocks onto PEs
 - Maps well on a 2D mesh
 - Finest granularity
 - 1 element per PE, ($p = n^2$)
- Many matrix algorithms allow block formulation
 - Matrix add
 - Matrix multiply

CS575 lecture 7

3

Matrix Transpose

for $i = 0$ *to* $n-1$

for $j = i+1$ *to* $n-1$

swap(A, i, j)

- Striped: (almost) all-to-all personal communication
- Checkerboard ($p = n^2$)
 - Upper triangle element travels down to diagonal then left
 - Lower triangle element travels up to diagonal then right
- Checkerboard ($p < n^2$)
 - Do above communication but with blocks: $2 \cdot \sqrt{p} \cdot (n^2/p)$ traffic
 - Transpose blocks at destination: $O(n^2/p)$ swaps

CS575 lecture 7

4

Matrix Vector Multiply - Striped

- $n \times n$ matrix A , $n \times 1$ vector x , $y = A.x$
- Row Striped, $p = n$
 - one row of A per PE
 - one element of x , y per PE
 - every PE needs all x 's
 - all-to-all broadcast: $O(n)$ time (ring, SF)
- Block row striped, $p < n$
 - n/p rows of A , n/p elements of x , y per PE
 - all-to-all broadcast of x blocks

CS575 lecture 7

5

Matrix Vector Multiply - Checkerboard

- Fine grain, Mesh
 - $p = n^2$, x in last column of mesh
 - Send x element to PE present on the diagonal: one-to-one
 - Broadcast x element along column: one-to-all BC
 - Multiply point-wise
 - Single node sum-reduction per row: (all-to-one)
 - e.g. into last column
- $p < n^2$
 - Do above algorithm for n/\sqrt{p} chunks of x , and $n/\sqrt{p} \times n/\sqrt{p}$ blocks of A
 - one-to-one: distance*size = $O(\sqrt{p} \times (n/\sqrt{p}))$
 - one-to-all: $O(\sqrt{p} \times (n/\sqrt{p}))$
 - Block multiply: rows * in-product: $n/(\sqrt{p}) \times n/(\sqrt{p})$
 - All to 1 reduction: $O(\sqrt{p} \times (n/\sqrt{p}))$

CS575 lecture 7

6

$n \times n$ Matrix Multiply

```
for i = 0 to n-1
  for j = 0 to n-1
    Cij = 0
    for k = 0 to n-1
      Cij += Aik * Bkj
```

We do not consider recursive $< O(n^3)$ algorithms
(s.a. Strassen)

CS575 lecture 7

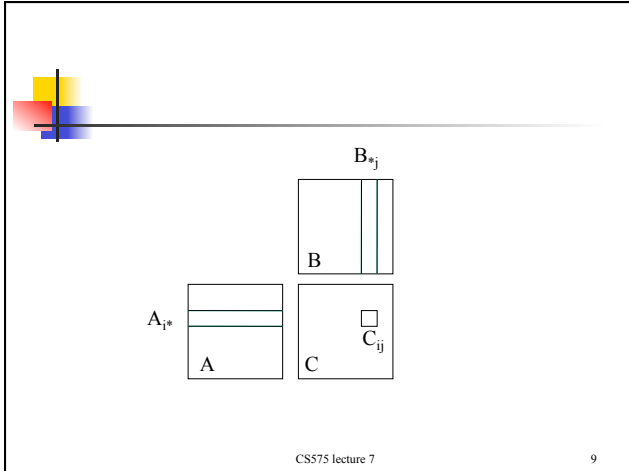
7

Blocked Matrix Multiply

- Standard algorithm can be easily blocked
- p processors: $n/\sqrt{p} \times n/\sqrt{p}$ sized blocks
- PE $_{ij}$ has blocks A_{ij} and B_{ij}
and computes block C_{ij}
- C_{ij} needs A_{ik} and B_{kj} , $k = 0$ to $n-1$
 - Assuming above initial data distribution, i.e. each data item is allocated in one memory, some form of communication is needed

CS575 lecture 7

8



- ### Simple Block Matrix Multiply
- All row PEs need complete rows of A
 - all-to-all block broadcast of A in row PEs
 - $O(\sqrt{p}) * (n/\sqrt{p}) * n/\sqrt{p})$
 - All column PEs need complete columns of B
 - all-to-all block broadcast of B in column PEs
 - $O(\sqrt{p}) * (n/\sqrt{p}) * n/\sqrt{p})$
 - Compute block C_{ij} in PE $_{ij}$: n^3/p
 - Space use EXCESSIVE:
 - per PE: $2 * \sqrt{p} * (n/\sqrt{p}) * n/\sqrt{p})$
 - Total: $2 * n * n * \sqrt{p}$
- CS575 lecture 7 10

- ### Cannon's Matrix Multiply
- Avoids space overhead
 - interleaves block moves and computation
 - PE $_{ij}$ computes block C_{ij}
 - Initial alignment of data
 - Circular left shift block A_{ij} by i steps
 - Circular up shift block B_{ij} by j steps
 - Interleave computation and communication
 - Compute: block matrix multiplication
 - Communicate:
 - circular shift left A blocks
 - circular shift up B blocks
- CS575 lecture 7 11

- ### Cost of Cannon's Matrix Multiply
- Initial data alignment
 - Aligning A or B
 - Worst distance * size $\approx \sqrt{p} * n^2/p$
 - Total $\approx 2 * \sqrt{p} * n^2/p$
 - Interleave computation and communication
 - Compute: total = n^3/p
 - Communicate:
 - A blocks circular shift left
 - B blocks circular shift up
 - Total cost = number of shifts * size $\approx 2 * \sqrt{p} * n^2/p$
 - Space: $2n^2/p$ per PE
- CS575 lecture 7 12

Fox's Matrix Multiply

- Avoids space overhead
 - interleaves broadcasts for A, block moves for B and computation
- Initial data distribution: standard block
 - Broadcast A_{ii} in row i
 - Compute: block matrix multiplication
- Do $j = 0$ to $\sqrt{p}-2$ times
 - Circular up shift B blocks
 - Broadcast A_{ik} block in row i , where $k = (j+1) \bmod \sqrt{p}$
 - Compute: block matrix multiplication

CS575 lecture 7

13

Cost of Fox's Matrix Multiply

- Broadcasts (one-to-all)
 - Volume = n^2/p
 - Distance (e.g.) = $\log(\sqrt{p})$ for mesh embedded on hypercube
- Computation: total = n^3/p
- $O(\sqrt{p})$ circular shifts
 - Each circular shift (nearest neighbor): volume = n^2/p

CS575 lecture 7

14

Dekel, Nassimi, Sahni Matrix Multiply

- Fine grain CREW PRAM formulation
 1. forall i, j, k : $C_{ij} = A_{ik} * B_{kj}$ // time: $O(1)$
 2. Sum reduce C_{ij} $k = 0 \dots n-1$ // time: $O(\log n)$ hyper cube, $O(n)$ 3D mesh
- 3D Mesh formulation: n^3 PEs, lots of data replication
 - plane corresponds to different values of k
 - A's columns distributed/replicated over X planes
 - B's rows distributed/replicated over Y planes
 - Do all point to point multiplies in parallel
 - Collapse sum reduction in Z planes
- This seems silly, but is pretty good in block form.

CS575 lecture 7

15

Solving Linear Equations: Gaussian Elimination

- Reduce $Ax=b$ into $Ux=y$
 - U is an upper triangular
 - Diagonal elements $U_{ii} = 1$

$$\begin{aligned} x_0 + u_{01} x_1 + \dots + u_{0\ n-1} x_{n-1} &= y_0 \\ x_1 + \dots + u_{1\ n-1} x_{n-1} &= y_1 \\ &\dots\dots\dots \\ x_{n-1} &= y_{n-1} \end{aligned}$$

- Back substitute

CS575 lecture 7

16

Upper Triangularization - Sequential

- Two phases repeated n times
- Consider the k-th iteration ($0 \leq k < n$)
- Phase 1: Normalize k-th row of A
 - for $j = k+1$ to $n-1$ $A_{kj} /= A_{kk}$
 - $y_k = b_k / A_{kk}$
 - $A_{kk} = 1$
- Phase 2: Eliminate
 - Using k-th row, make k-th column of A zero for row# $> k$
 - for $i = k+1$ to $n-1$
 - for $j = k+1$ to $n-1$ $A_{ij} -= A_{ik} * A_{kj}$
 - $b_i -= a_{ik} * y_k$
 - $A_{ik} = 0$
- $O(n^2)$ divides, $O(n^3)$ subtracts and multiplies

CS575 lecture 7

17

Upper Triangularization - Parallel

- $p = n$, row-striped partition
- for $k = 0$ to $n-1$
- k-th phase 1: normalize
 - performed by P_k
 - sequentially, no communication
- k-th phase 2: eliminate
 - P_k broadcasts k-th row to P_{k+1}, \dots, P_{n-1}
 - performed in parallel by P_{k+1}, \dots, P_{n-1}

CS575 lecture 7

18

Upper Triangularization – Pipelined Parallel

- $p = n$, row-stripes partition
- for all P_i ($i = 0 \dots n-1$) do in parallel
- for $k = 0$ to $n-1$
- if ($i == k$)
 - perform k-th phase 1: normalize
 - send normalized row k down
- if ($i > k$)
 - receive row k, send it down
 - perform k-th phase 2: eliminate with row k

CS575 lecture 7


19

Pivoting in Gaussian elimination

- What if $A_{kk} \sim 0$?
 - We get a big error
- Find a lower row e with largest A_{ek} and exchange rows
- What is all $A_{ek} \sim 0$?
 - Find a column with largest A_{ke} and exchange columns
- This complicates the parallel algorithm

CS575 lecture 7

20



Back-substitute – Pipelined row striped

$$\begin{aligned}x_0 + u_{0,1} x_1 + \dots + u_{0,n-1} x_{n-1} &= y_0 \\x_1 + \dots + u_{1,n-1} x_{n-1} &= y_1 \\&\dots\dots\dots \\x_{n-1} &= y_{n-1}\end{aligned}$$

for $k = n-1$ down to 0

P_k : $x_k = y_k$; send x_k up

$P_{i(i < k)}$: send x_k up, $y_i = y_i - x_k * u_{ik}$