JOHANNES KEPLER
UNIVERSITY LINZ

Stefan Amberger

ICA & RISC
amberger.stefan@gmail.com

A Parallel, In-Place, Rectangular Matrix Transpose Algorithm

Computational Complexity Analysis

Table of Contents

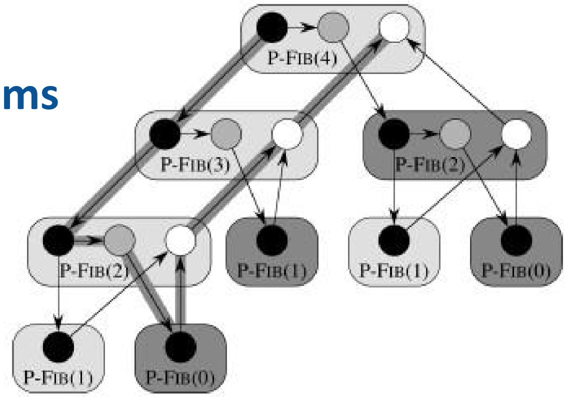
1. Introduction
2. Revision of TRIP
3. Analysis of Computational Complexity
    a. Work
    b. Span
    c. Parallelism
    d. Generalizations

JOHANNES KEPLER
UNIVERSITY LINZ

Introduction

Introduction

Computational Complexity of Parallel Algorithms



Introduction to Algorithms Third Edition, p777ff

Work W_1

"execution time on one processor"

i.e.: **all vertices** of computation dag

approximation: # of nodes of computation dag

Span W_∞

"execution time on infinitely many processors"

i.e. length of **critical path** of computation dag

approximation: # of nodes on critical path

Parallelism $\frac{W_1}{W_\infty}$

- average amount of work per step along critical path
- maximum possible speedup
- limit on possibility of attaining perfect speedup

Revision of TRIP

JOHANNES KEPLER
UNIVERSITY LINZ

TRIP:
If matrix is rectangular ***TRIP*** transposes sub-matrices, then combines the result with ***merge*** or ***split***

merge:
first rotates the middle part of the array, then recursively merges the left and right parts of the array

split:
first recursively splits the left and right parts of the array, then rotates the middle part of the array

JOHANNES KEPLER
UNIVERSITY LINZ

Analysis of Computational Complexity

Restriction to Powers of Two

"power condition"

Matrix dimensions M x N and N x M

$$\left(\exists k \in \mathbb{N} : N = 2^k\right) \wedge \left(\exists l \in \mathbb{N} : M = 2^l N\right)$$

recursive calls are symmetric

TRIP's recursive call are either all `merge` or all `split`

Work

1. Example: Basic Algorithms
2. TRIP Result & Proof Sketch

Work Example

Work of Base Algorithms

```
cilk reverse(A, m0, m1, l) {
  if (l > 1) {
    lm = l/2;
    spawn reverse(A, m0, m1, lm);
    spawn reverse(A, m0 + lm, m1 - lm, l - lm);
  } else {
    swap(A, m0, m1 - 1);
  }
}
```

Lemma 1 (Work of reverse) *If length $n = m_1 - m_0$ of array A is even, then*

$$W_1^{reverse}(n) = n - 1$$

Proof. initially $l = n/2$, base case $l = 1$

binary tree has $n/2 - 1$ nodes

1 swap per leaf: $n/2$ swaps

$$W_1^{\text{reverse}}(n) = \frac{n}{2} - 1 + \frac{n}{2} = n - 1$$

□

```
cilk rol(A, n, k) {
  spawn reverse(a, 0, k);
  spawn reverse(a, k, n);
  sync;
  spawn reverse(a, 0, n);
}
```

Lemma 2 (Work of rol) *If length n of array A is even, then the work of rol$(n,\ n/2)$ is*

$$W_1^{rol}(n,\ n/2) = 2n - 3$$

Proof. algorithm not recursive

half array rotations: work $n/2 - 1$ each
full array rotation: work $n - 1$

$$W_1^{\text{rol}}(n,\ n/2) = 2\left(\frac{n}{2} - 1\right) + (n - 1) = 2n - 3$$

□

Result

Work of TRIP

Show that under power condition, for M x N matrix

$$W_1^{\mathrm{TRIP}}(M,\ N) = \Theta\left(MN\left(1 + \log\frac{M}{N}\ \log\ N\right)\right)$$

and in general:

$$W_1^{\mathrm{TRIP}}(M,\ N) = \Theta\left(MN\left(1 + \log\frac{\max(M,\ N)}{\min(M,\ N)}\ \log\ \min(M,\ N)\right)\right)$$

METHOD

don't count vertices in computation dag

count **inner nodes in recursion tree** and **swaps**

- function calls (nodes in recursive call trees)
- memory accesses (swaps)

Work of TRIP

Proof Sketch

Work of TRIP

$$W_1^{\mathrm{TRIP}}(M,\ N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le i < \lg \frac{M}{N}} 2^i W_1^{\mathrm{merge}}(p_i,\ q_i,\ N)}_{\text{combine effort}} + \underbrace{\frac{M}{N} W_1^{\mathrm{square}}(N)}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

Work of merge

$$W_1^{\mathrm{merge}}(p_i,\ q_i,\ N) = \underbrace{N - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le j < \lg N} 2^j W_1^{\mathrm{rol}}\left(\frac{n_j}{2}p_i + \frac{n_j}{2}q_i,\ \frac{n_j}{2}p_i\right)}_{\text{work within inner nodes of the merge tree}}$$

Work of rol

$$W_1^{\mathrm{rol}}\left(\frac{n_j}{2}p_i + \frac{n_j}{2}q_i,\ \frac{n_j}{2}p_i\right) = NM2^{-i-j} - 3$$

Work of TRIP

wide matrices

TRIP recursion is analogous for tall and wide matrices

only difference:

- in `merge` `rol` is called before the recursive merge call
- in `splitt` `rol` is called after the recursive split call

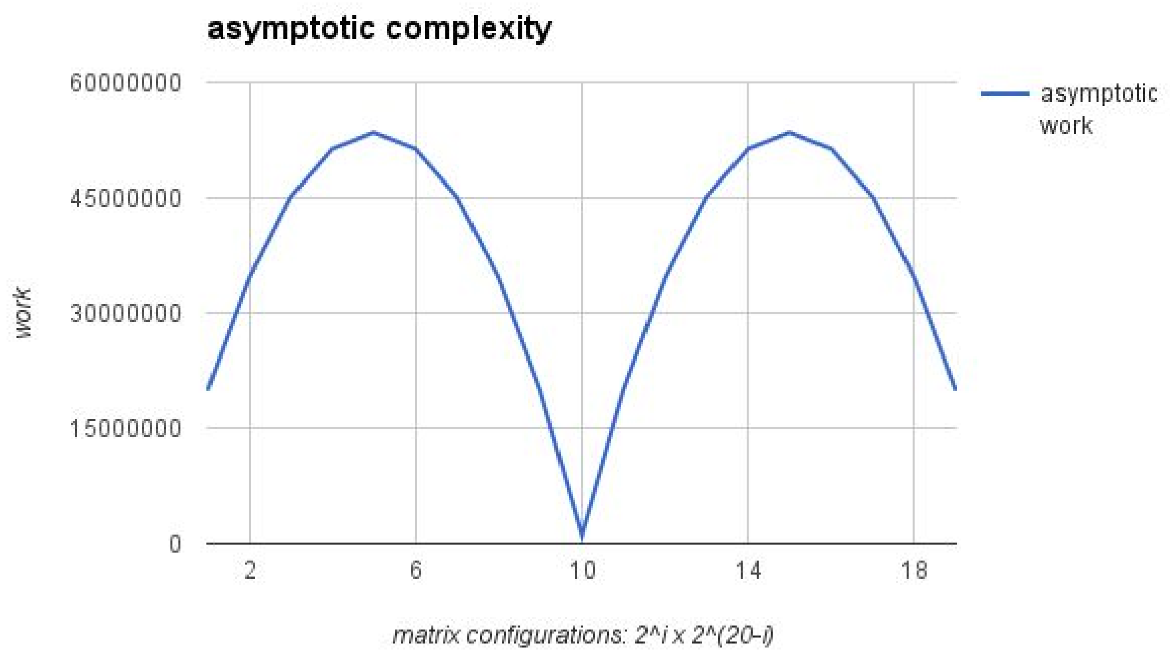
This difference does not cause a change in the amount of work of `TRIP`.

Corollary 1 (Work of TRIP). *Let $A_{M,N}$ be a tall, wide or square matrix that satisfies the power condition. Then*

$$W_1^{TRIP}(M,\ N) = \Theta\left(MN\left(1 + \log\frac{\max(M,\ N)}{\min(M,\ N)}\ \log\ \min(M,\ N)\right)\right)$$

Visualization

Work as function of Matrix Dimensions



Span

1. Example: Basic Algorithms
2. TRIP Result

JOHANNES KEPLER UNIVERSITY LINZ

Span Example

Span of Base Algorithms

```
cilk reverse(A, m0, m1, l) {
  if (l > 1) {
    lm = l/2;
    spawn reverse(A, m0, m1, lm);
    spawn reverse(A, m0 + lm, m1 − lm, l − lm);
  } else {
    swap(A, m0, m1 − 1);
  }
}
```

Lemma 3 (Span of **reverse**). *If length n of array A is a power of two, then*

$$W_\infty^{reverse}(n) = \log\ \frac{n}{2} + 1$$

Proof. initially $l = n/2$, base case $l = 1$

binary tree has log $n/2$ levels

1 swap in leaf adds 1 to span

$$W_\infty^{\texttt{reverse}}(n) = \log\ \frac{n}{2} + 1$$

□

```
cilk rol(A, n, k) {
  spawn reverse(a, 0, k);
  spawn reverse(a, k, n);
  sync;
  spawn reverse(a, 0, n);
}
```

Lemma 4 (Span of `rol`). *If length n of array A is a power of two, then the span of rol$(n,\ n/2)$ is*

$$W_\infty^{rol}(n) = \log\ 2^{-1}n + \log\ 2^{-2}n + 3$$

Proof. algorithm not recursive

`rol` consists of three reversals of lengths $n/2$, $n/2$ and n
first two are in parallel
Lemma 3
adding span 1 for high-level function calls (cf. inner nodes)

$$W_\infty^{\mathrm{rol}}(n,\ n/2) = \log\ \frac{n}{2} + \log\ \frac{n}{4} + 3 = \log\ 2^{-1}n + \log\ 2^{-2}n + 3$$

□

Span of TRIP

Result

Calculate span of tall matrix transpose

count **levels** and **swaps** on critical path, that includes

span of

- creating the divide tree
- combining the nodes via `merge`/`split` (itself recursive procedures)
- square-transposing in the leaf nodes

Theorem 1 (Span of TRIP for tall matrices)**.** *Let $A_{M,N}$ be a* ***tall*** *matrix that satisfies the power condition.*

Then

$$W_{\infty}^{TRIP}(M,\ N) = \log\ \frac{M}{N} + \log\ \frac{M}{N}\ \log\ N\ \left(3\ \log\ N + \log\ \frac{M}{N}\right) + \log\ N + 1$$

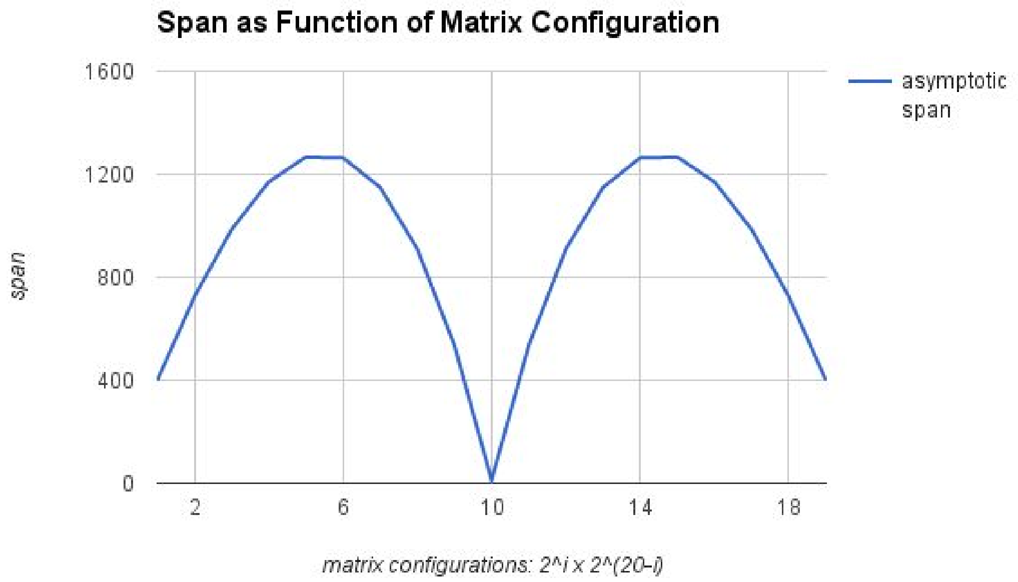
in general

$$W_{\infty}^{\mathrm{TRIP}}(M,\ N) = \log\ \frac{m}{n} + \log\ \frac{m}{n}\ \log\ n\ \left(3\ \log\ n + \log\ \frac{m}{n}\right) + \log\ n + 1$$

where $m = \max(M,\ N)$ and $n = \min(M,\ N)$

Visualization

Span as function of Matrix Dimensions



Parallelism

Result

Rectangular Matrices

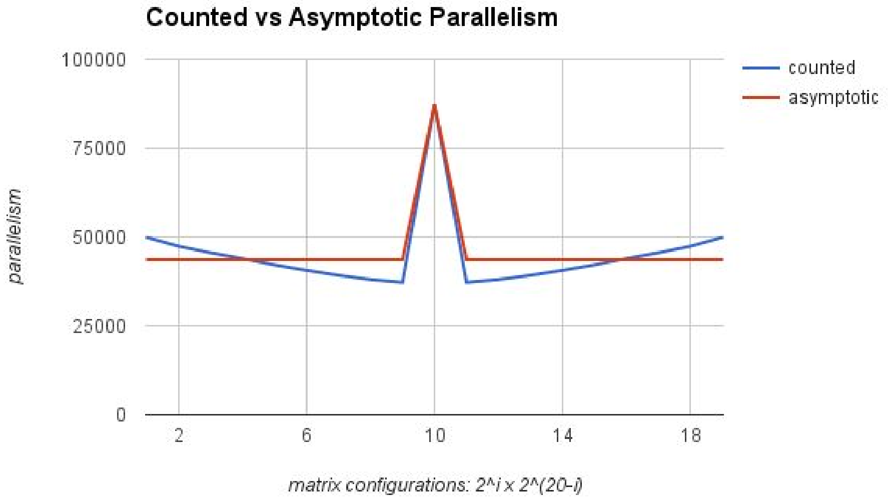
$$\Theta\left(\frac{MN}{\log\ M/N+\log\ N}\right)$$

Square Matrices

$$\Theta\left(\frac{N^2}{\log\ N}\right)$$

calculation:

- divide work by span
- case distinction rectangular / square
- simplification using Landau symbols



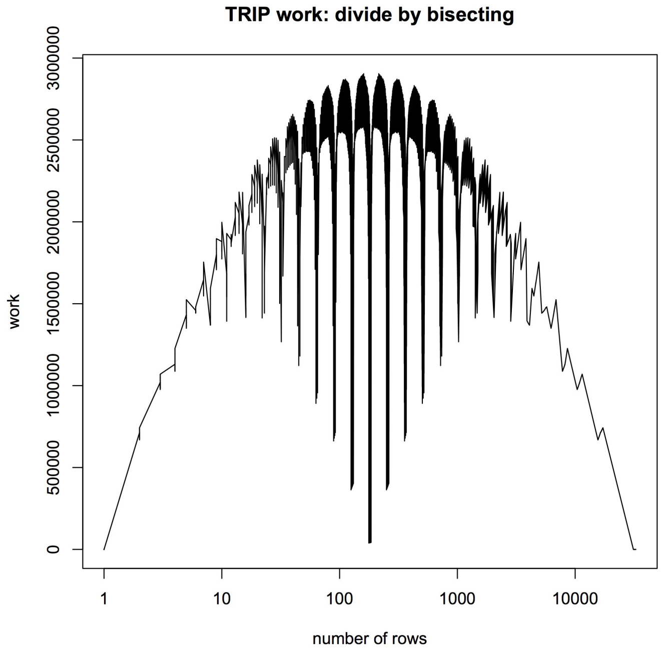
Generalizations

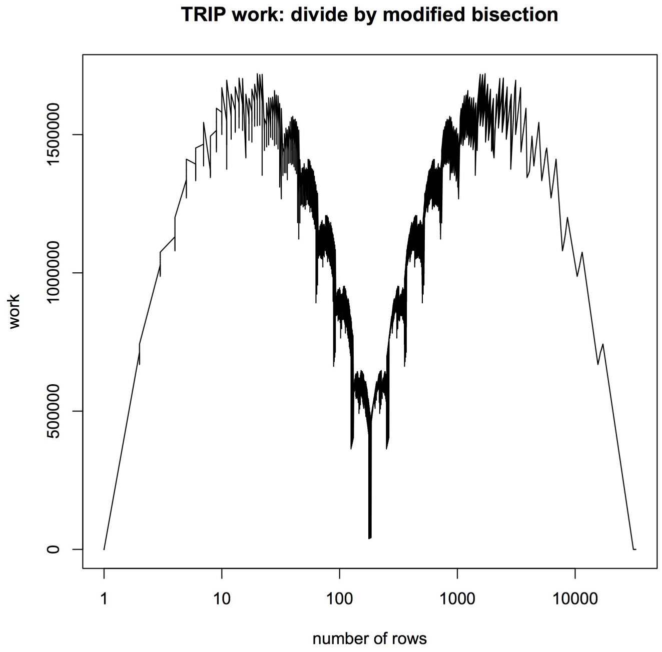
power condition unsatisfied

Example: 7 x 5 Matrix

Generalization

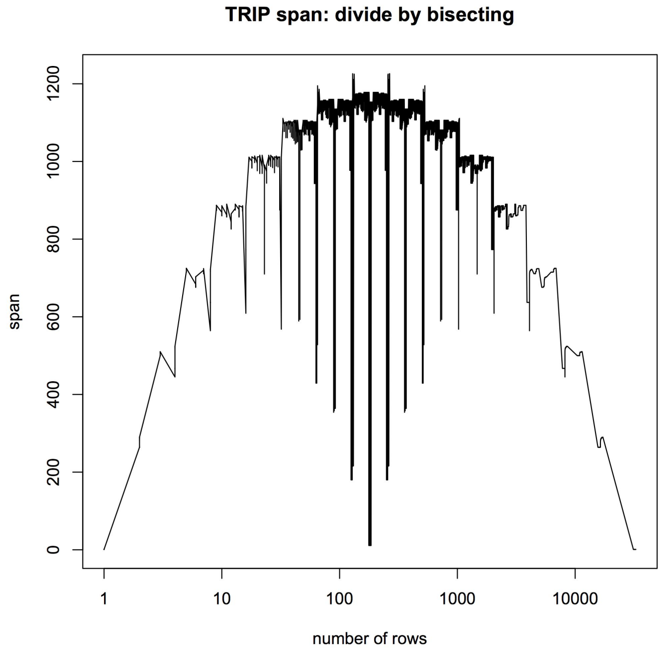
Power Condition not Satisfied

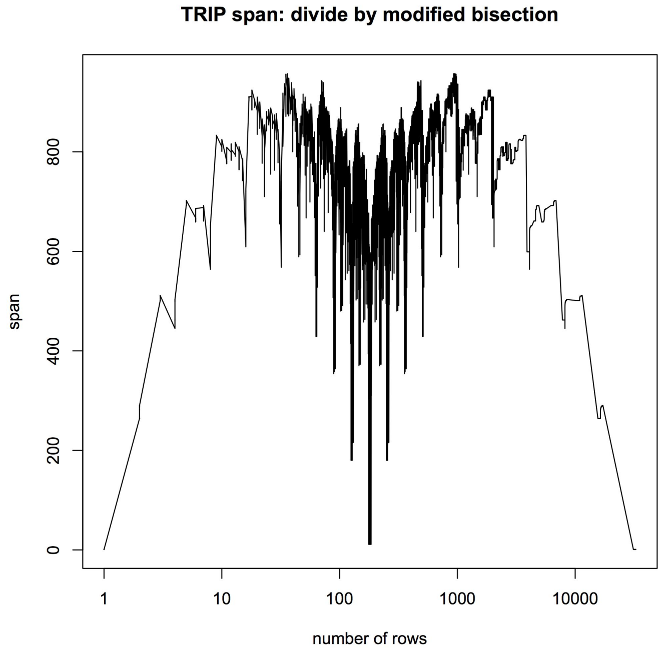




Generalization

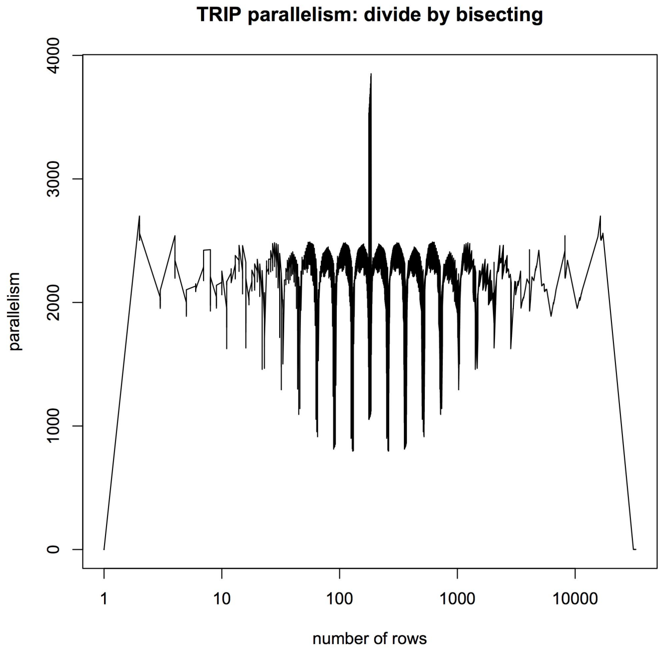
Power Condition not Satisfied

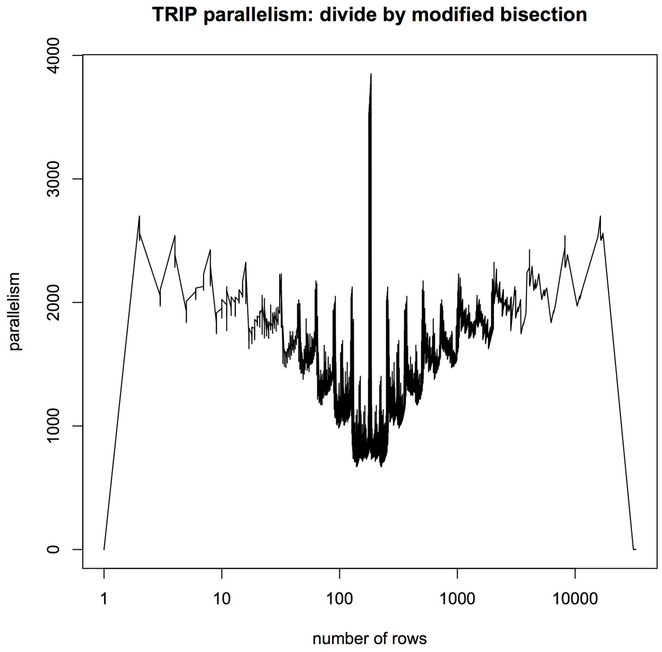




Generalization

Power Condition not Satisfied





Thank you

JOHANNES KEPLER
UNIVERSITY LINZ

Revision of TRIP

TRIP Algorithm

If matrix is rectangular ***TRIP*** transposes sub-matrices, then combines the result with ***merge*** or ***split***

$$
\mathrm{TRIP}(A,\ m,\ n) = \begin{cases}
\begin{array}{l}\mathrm{TRIP}(A(0:\lfloor \frac{m}{2} \rfloor,\ 0:n),\ \lfloor \frac{m}{2} \rfloor,\ n) \parallel \\ \mathrm{TRIP}(A(\lfloor \frac{m}{2} \rfloor : m,\ 0:n),\ \lceil \frac{m}{2} \rceil,\ n); \\ \mathrm{merge}(\overline{A},\ \lfloor \frac{m}{2} \rfloor,\ \lceil \frac{m}{2} \rceil,\ n)\end{array} & \text{if } m > n \\
\\
\begin{array}{l}\mathrm{TRIP}(A(0:m,\ 0:\lfloor \frac{n}{2} \rfloor),\ m,\ \lfloor \frac{n}{2} \rfloor) \parallel \\ \mathrm{TRIP}(A(0:m,\ \lfloor \frac{n}{2} \rfloor : n),\ m,\ \lceil \frac{n}{2} \rceil); \\ \mathrm{split}(\overline{A},\ \lfloor \frac{n}{2} \rfloor,\ \lceil \frac{n}{2} \rceil,\ m)\end{array} & \text{if } m < n \\
\\
\mathrm{square\_transpose}(A,\ n) & \text{if } m = n
\end{cases}
$$

`merge` Algorithm

merge combines the transposes of sub-matrices of *tall* matrices

merge first rotates the middle part of the array, then recursively merges the left and right parts of the array

$$\operatorname{merge}(\overline{A},\ p,\ q,\ n) = \begin{cases} \operatorname{rol}(\overline{A}(\lfloor \frac{n}{2} \rfloor p : np + \lfloor \frac{n}{2} \rfloor q),\ \lceil \frac{n}{2} \rceil p); & \\ \operatorname{merge}(\overline{A}(0 : \lfloor \frac{n}{2} \rfloor (p+q)),\ p,\ q,\ \lfloor \frac{n}{2} \rfloor) \parallel & \text{if } n > 1 \\ \operatorname{merge}(\overline{A}(\lfloor \frac{n}{2} \rfloor (p+q) : n(p+q)),\ p,\ q,\ \lceil \frac{n}{2} \rceil) & \\ & \\ \overline{A} & \text{if } n = 1 \end{cases}$$

rol(arr, k) … left rotation (circular shift) of array ***arr*** by ***k*** elements

`split` Algorithm

split combines the transposes of sub-matrices of *wide* matrices

split first recursively splits the left and right parts of the array, then rotates the middle part of the array

$$\operatorname{split}(\overline{A},\ p,\ q,\ m) = \begin{cases} \operatorname{split}(\overline{A}(0 : \lfloor \frac{m}{2} \rfloor (p+q)),\ p,\ q,\ \lfloor \frac{m}{2} \rfloor) \parallel \\ \operatorname{split}(\overline{A}(\lfloor \frac{m}{2} \rfloor (p+q) : m(p+q)),\ p,\ q,\ \lceil \frac{m}{2} \rceil); & \text{if } \ m > 1 \\ \operatorname{rol}(\overline{A}(\lfloor \frac{m}{2} \rfloor p : mp + \lfloor \frac{m}{2} \rfloor q),\ \lfloor \frac{m}{2} \rfloor q) & \\ \\ \overline{A} & \text{if } \ m = 1 \end{cases}$$

split and ***merge*** are inverse to each other

Work Proof

Work of TRIP

Overview

Calculate work of tall matrix transpose

- spanning the divide tree
- combining the nodes via `merge`/`split` (itself recursive procedures)
- square-transposing in the leaf nodes

Theorem 1 (Work of `TRIP` for tall matrices). *Let $A_{M,N}$ be a **tall** matrix that satisfies the power-condition.*

Then

$$W_1^{\mathit{TRIP}}(M,\ N) = \Theta\left(MN\left(1 + \log\frac{M}{N}\ \log\ N\right)\right)$$

Work of TRIP

Proof - TRIP Tree

Spanning Divide Tree

recursion parameter of transpose is m, $m_0 = M$

$$\forall_{0 \le i < \lg \frac{M}{N}} m_{i+1} = m_i/2$$

base case is $m = N$ (square sub-matrix)

- $\lg \frac{M}{N}$ levels
- $\frac{M}{N}$ leafs
- $\frac{M}{N} - 1$ inner nodes

Combining Nodes via `merge`

number of inner nodes at level i is 2^i

at level i $\texttt{merge}(m_i/2,\ m_i/2,\ N)$ is called

$m_0 = M \Rightarrow m_i/2 = M2^{-(i+1)}$

$$W_1^{\text{TRIP}}(M,\ N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le i < \lg \frac{M}{N}} 2^i \boxed{W_1^{\text{merge}}(p_i,\ q_i,\ N)}}_{\text{combine effort}} + \underbrace{\frac{M}{N} W_1^{\text{square}}(N)}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

Work of TRIP

Proof - Merge Tree

Combining via `merge`, rotate sub-arrays

$$W_1^{\text{TRIP}}(M,\ N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le i < \lg \frac{M}{N}} 2^i \boxed{W_1^{\text{merge}}(p_i,\ q_i,\ N)}}_{\text{combine effort}} + \underbrace{\frac{M}{N} W_1^{\text{square}}(N)}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

recursion parameter n with $n_0 = N$

base case $n_j = 1$ for some j

$n_{j+1} = n_j/2$

$\Rightarrow n_j = N2^{-j}$ for $0 \le j < \log\ N$.

$$W_1^{\text{merge}}(p_i,\ q_i,\ N) = \underbrace{N-1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le j < \lg\ N} 2^j W_1^{\text{rol}}\left(\frac{n_j}{2}p_i + \frac{n_j}{2}q_i,\ \frac{n_j}{2}p_i\right)}_{\text{work within inner nodes of the merge tree}}$$

- `merge` call has $\log\ N$ levels
- $N-1$ inner nodes
- 2^j inner nodes at level j of merge tree
- 1 `rol` call per inner node

$n_j = N2^{-j}$ for $0 \le j < \log\ N$

$p_i = q_i = m_i/2 = M2^{-(i+1)}$

Lemma 2

$$W_1^{\text{rol}}\left(\frac{n_j}{2}p_i + \frac{n_j}{2}q_i,\ \frac{n_j}{2}p_i\right) = W_1^{\text{rol}}\left(NM2^{-i-j-1},\ \frac{1}{2}NM2^{-i-j-1}\right)$$
$$= NM2^{-i-j} - 3$$

Work of TRIP

Proof - Merge Tree

$$W_1^{\text{TRIP}}(M,\ N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0\le i<\lg\frac{M}{N}} 2^i \boxed{W_1^{\text{merge}}(p_i,\ q_i,\ N)}}_{\text{combine effort}} + \underbrace{\frac{M}{N} W_1^{\text{square}}(N)}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

Integrate `rol` result into `merge` work

$$W_1^{\text{merge}}(p_i,\ q_i,\ N) = \underbrace{N-1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0\le j<\lg N} 2^j W_1^{\text{rol}}\left(\frac{n_j}{2}p_i + \frac{n_j}{2}q_i,\ \frac{n_j}{2}p_i\right)}_{\text{work within inner nodes of the merge tree}}$$

$$W_1^{\text{rol}}\left(\frac{n_j}{2}p_i + \frac{n_j}{2}q_i,\ \frac{n_j}{2}p_i\right) = NM2^{-i-j} - 3$$

$$\begin{aligned}
W_1^{\text{merge}}(p_i,\ q_i,\ N) &= N-1+\sum_{0\le j<\log N} 2^j W_1\left(\text{rol}\left(\frac{n_j}{2}p_i + \frac{n_j}{2}q_i,\ \frac{n_j}{2}p_i\right)\right)\\
&= N-1+\sum_{0\le j<\log N} 2^j\left(NM2^{-i-j}-3\right)\\
&= N-1+\log(N)NM2^{-i}-3\sum_{0\le j<\log N} 2^j\\
&= N-1+\log(N)NM2^{-i}-3(N-1)\\
&= \log(N)NM2^{-i}-2(N-1)
\end{aligned}$$

Work of TRIP

Proof - TRIP Tree

$$W_1^{\text{TRIP}}(M,\ N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le i < \lg \frac{M}{N}} 2^i W_1^{\text{merge}}(p_i,\ q_i,\ N)}_{\text{combine effort}}$$
$$+ \underbrace{\frac{M}{N} \boxed{W_1^{\text{square}}(N)}}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

Integrate merge result into TRIP work

$$W_1^{\text{merge}}(p_i,\ q_i,\ N) = \log\ (N) N M 2^{-i} - 2(N-1)$$

$$W_1^{\text{TRIP}}(M,\ N) = (-2N+3)\left(\frac{M}{N} - 1\right) + \left(MN\ \lg\ \frac{M}{N}\ \lg\ N\right) + \frac{M}{N} W_1^{\text{square}}(N)$$

$$\begin{aligned}
W_1^{\text{TRIP}}(M,\ N) &= \left(\frac{M}{N} - 1\right) + \sum_{0 \le i < \lg \frac{M}{N}} 2^i W_1^{\text{merge}}(p_i,\ q_i,\ N) + \frac{M}{N} W_1^{\text{square}}(N) \qquad (1)\\
&= \left(\frac{M}{N} - 1\right) + \sum_{0 \le i < \lg \frac{M}{N}} 2^i\ \left(\lg\ (N) N M 2^{-i} - 2(N-1)\right) + \frac{M}{N} W_1^{\text{square}}(N)\\
&= \left(\frac{M}{N} - 1\right) + \left(\lg\ \frac{M}{N}\ \lg\ (N) N M - 2(N-1)\left(\frac{M}{N} - 1\right)\right) + \frac{M}{N} W_1^{\text{square}}(N)\\
&= (-2N+3)\left(\frac{M}{N} - 1\right) + \left(MN\ \lg\ \frac{M}{N}\ \lg\ N\right) + \frac{M}{N} W_1^{\text{square}}(N)
\end{aligned}$$

Work of TRIP

Proof - Square Transpose

$$W_1^{\mathrm{TRIP}}(M, N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le i < \lg \frac{M}{N}} 2^i W_1^{\mathrm{merge}}(p_i, q_i, N)}_{\text{combine effort}} + \underbrace{\frac{M}{N} \boxed{W_1^{\mathrm{square}}(N)}}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

Recap

```
cilk square_transpose(A, i0, i1, j0, j1) {
  if (i1 − i0 > 1) {
    im = (i0 + i1)/2;
    jm = (j0 + j1)/2;
    spawn square_transpose(A, i0, im, j0, jm);
    spawn square_transpose(A, i0, im, jm, j1);
    spawn square_transpose(A, im, i1, jm, j1);
    if (i1 ≤ j0)
      spawn square_transpose(A, im, i1, j0, jm);
  } else {
    for (j = j0; j < j1; j++) {
      swap(A[j, i0], A[i0, j]);
    }
  }
}
```

Work of TRIP

Proof - Square Transpose

$$W_1^{\text{TRIP}}(M,\ N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0\le i<\lg\ \frac{M}{N}} 2^i W_1^{\text{merge}}(p_i,\ q_i,\ N)}_{\text{combine effort}} + \underbrace{\frac{M}{N} \boxed{W_1^{\text{square}}(N)}}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

tree covers upper right matrix: $N(N+1)/2$ leaves

Lower Bound on # of inner nodes

purely ternary tree

tree would have $\lceil \log_3(N(N+1)/2) \rceil$ levels

number of inner nodes:

$$\sum_{i=0}^{\lceil \log_3(N(N+1)/2)-1 \rceil} 3^i$$

Since

$$\sum_{k=0}^{m} 3^i = \frac{1}{2}\left(3^{m+1} - 1\right)$$

Number of inner nodes is about

$$\sum_{i=0}^{\log_3(N(N+1)/2)-1} 3^i = \frac{1}{2}\left(3^{\log_3(N(N+1)/2)} - 1\right) = \frac{N(N+1)}{4} - \frac{1}{2} = \Theta(N^2)$$

Upper Bound on # of inner nodes

purely quaternary tree

tree would have $\lceil \log_4(N(N+1)/2) \rceil$ levels

number of inner nodes:

$$\sum_{i=0}^{\lceil \log_4(N(N+1)/2)-1 \rceil} 4^i$$

Since

$$\sum_{k=0}^{m} 4^i = \frac{1}{3}\left(4^{m+1} - 1\right)$$

Number of inner nodes is about

$$\sum_{i=0}^{\log_4(N(N+1)/2)-1} 4^i = \frac{1}{3}\left(4^{\log_4(N(N+1)/2)} - 1\right) = \frac{N(N+1)}{6} - \frac{1}{3} = \Theta(N^2)$$

Work of TRIP

Proof - TRIP Tree

$$W_1^{\text{TRIP}}(M,\ N) = \underbrace{\frac{M}{N} - 1}_{\#\text{ of inner nodes}} + \underbrace{\sum_{0 \le i < \lg\ \frac{M}{N}} 2^i W_1^{\text{merge}}(p_i,\ q_i,\ N)}_{\text{combine effort}} + \underbrace{\frac{M}{N} W_1^{\text{square}}(N)}_{\text{work in leaves of transpose tree (square\_transpose)}}$$

Integrate `square` `transpose` result into TRIP work

work of square transpose (including swapping)

$$W_1^{\text{square}}(N) = \Theta(N^2) + N(N-1)/2 = \Theta(N^2)$$

$$\begin{aligned} W_1^{\text{TRIP}}(M,\ N) &= (-2N+3)\left(\frac{M}{N} - 1\right) + \left(MN\ \lg\ \frac{M}{N}\ \lg\ N\right) + \frac{M}{N} W_1^{\text{square}}(N) \\ &= (-2N+3)\left(\frac{M}{N} - 1\right) + \left(MN\ \lg\ \frac{M}{N}\ \lg\ N\right) + \frac{M}{N}\Theta(N^2) \end{aligned}$$

Which simplifies to

$$\begin{aligned} W_1^{\text{TRIP}}(M,\ N) &= \Theta\left(M + MN\ \log\ \frac{M}{N}\ \log\ N + MN\right) \\ &= \Theta\left(MN\left(1 + \log\ \frac{M}{N}\ \log\ N\right)\right) \end{aligned}$$

JOHANNES KEPLER
UNIVERSITY LINZ

Conclusions

Conclusions

Novel Algorithm ***TRIP*** transposes rectangular matrices

- correctly
- in-place
- in highly parallel manner

Roadmap

1. Work
2. Span
3. Parallelism