Scalability of Parallel Algorithms for Matrix Multiplication

Presented by: Abhishek Kumar

Gupta, Anshul, and Vipin Kumar. "Scalability of parallel algorithms for matrix multiplication." *1993 International Conference on Parallel Processing-ICPP'93*. Vol. 3. IEEE, 1993.

Motivation

- Analyze the performance and scalability of a number of parallel formulations of the matrix multiplication algorithm
- Predict the conditions under which each formulation is better than the other

Introduction

- Matrix multiplication is used in a variety of application
- Matrix multiplication formulations:
  - Cannon’s algorithm
  - Berntsen’s algorithm
  - DNS algorithm
- Near linear speedups for sufficiently large matrices
- Isoefficiency metric to analyze the scalability

Cannon’s Algorithm

- Two n x n matrices A and B are divided into square submatrices of size $\frac{n}{\sqrt{p}}$ x $\frac{n}{\sqrt{p}}$ among p processors.
- Data from block A^{ij} is sent to processor (i, (j+i)mod$\sqrt{p}$). Similarly, block B^{ij} sends data to processor ((i+j)mod$\sqrt{p}$, j).
- Sub-blocks of A are rolled one step left and B sub-blocks are rolled one step up and multiplied.
- Multiplication of A and B is complete after $\sqrt{p}$ steps.

Cannon’s Algorithm

Total parallel execution time: $\frac{n^3}{p} + 2t_s\sqrt{p} + 2t_w \frac{n^2}{\sqrt{p}}$

Berntsen’s Algorithm

- Berntsen proposed a algorithm which exploits greater connectivity of the hypercube.
- p = 2^{3q} processors with $p \leq n^{3/2}$ restriction
- Matrix A is split by columns and matrix B by rows into 2^q parts
- Hypercube is split into 2q sub-cubes, each performing a submatrix multiplication between submatrices A ($\frac{n}{2^{2q}}$ x $\frac{n}{2^{2q}}$) and B ($\frac{n}{2^{2q}}$ x $\frac{n}{2^{2q}}$) using Cannon’s algorithm

Berntsen’s Algorithm

Total parallel execution time: $\frac{n^3}{p} + 2t_s\, p^{1/3} + \frac{1}{3} t_s \log p + 3t_w \frac{n^2}{p^{2/3}}$

- Terms associated with both t_s and t_w are smaller than Cannon’s algorithm

DNS Algorithm

- Initially, $p = n^3 = 2^{3q}$ processors
- Completed the task of $O(n^3)$ matrix multiplication in $O(\log n)$ time using n^3 processors
- A proposed variant could work with n^2r processors $(1 < r < n)$
- Logical processor array of r^3 superprocessors is used, each comprising of $(n/r)^2$ hypercube processors
- Multiplication of $(n/r) \times (n/r)$ blocks is performed on $\frac{n}{r} \times \frac{n}{r}$ subarrays of processors using Cannon's algorithm

GK Algorithm

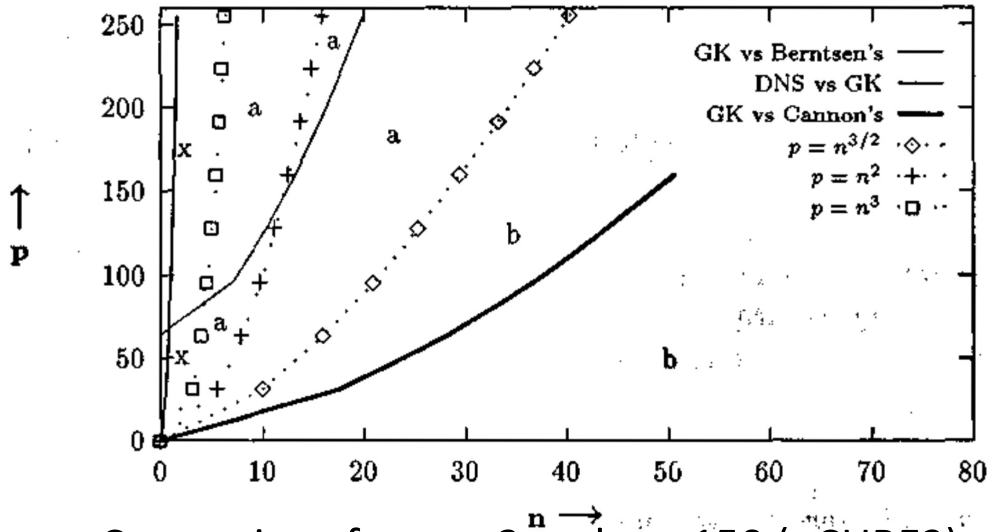
- Another scheme to adapt the DNS algorithm to use fewer than n^3 processors
- $p = 2^{3q}$ processors with $q < \frac{1}{3} \log n$
- Matrices are divided into sub-blocks of $\frac{n}{2^{2q}} \times \frac{n}{2^{2q}}$ elements
- In this variant of the DNS algorithm, all the single element operations are replaced by sub-block operations
- Total parallel execution time: $\frac{5}{3} t_s \, p \log p + \frac{5}{3} t_w \, n^2 \, p^{1/3} \log p$

Comparison of various algorithms

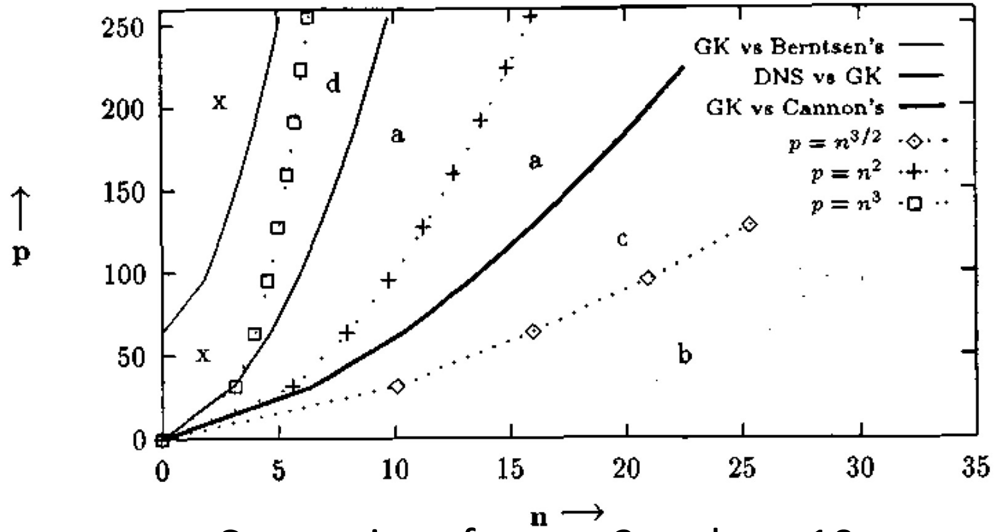
Algorithm	Total Overhead function	Asymptotic Isoefficiency	Applicability range
Cannon's	$2t_s\sqrt{p} + 2t_w \frac{n^2}{\sqrt{p}}$	$O(p^{1.5})$	$1 \le p \le n^2$
Berntsen's	$2t_s\, p^{1/3} + \frac{1}{3} t_s \log p + 3t_w \frac{n^2}{p^{2/3}}$	$O(p^2)$	$1 \le p \le n^{3/2}$
DNS	$(t_s + t_w)\, (\frac{5}{3} p \log p + 2n^3)$	$O(p \log p)$	$n^2 \le p \le n^3$
GK	$\frac{5}{3} t_s\, p \log p + \frac{5}{3} t_w\, n^2\, p^{1/3} \log p$	$O(p\, (\log p)^3)$	$1 \le p \le n^3$

- Only asymptotic scalabilities
- None of the algorithms is strictly better than others for all possible problem sizes and number of processors
- Compare pairwise the total overhead functions – GK vs Cannon's:
  - t_s term for GK is always smaller than Cannon's
  - Even if t_s=0, the t_w for GK becomes smaller than Cannon's for p > 130 million (irrespective of n)
  - For reasonable values of t_s, GK performs better than Cannon's for very practical values of p and n
  - n = $\sqrt{\frac{(5/3p \log p - 2\, p^{3/2}) t_s}{(2\sqrt{p} - 5/3 p^{1/3} \log p) t_w}}$

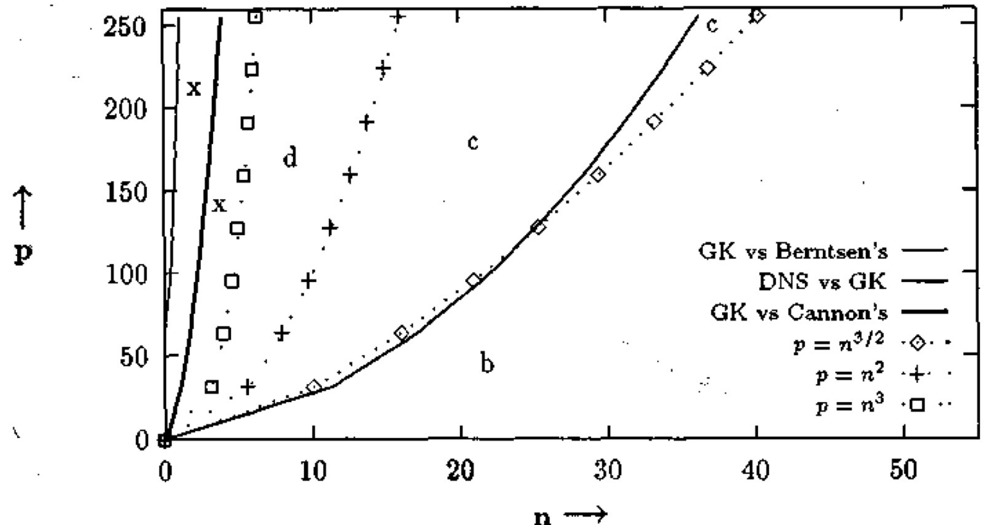
Comparison of various algorithms



Comparison for $t_w = 3$ and $t_s = 150$ (nCUBE2)



Comparison for $t_w = 3$ and $t_s = 10$



Comparison for $t_w = 3$ and $t_s = 0.5$ (SIMD machine)

- Regions marked ‘x’ is where non of the algorithms apply and $p>n^3$
- Region ‘a’ (GK), ‘b’ (Brentsen’s), ‘c’ (Cannon’s), ‘d’ (DNS algorithm)

Conclusion

- Scalability analysis provides insights into relative superiority under different conditions
- Predict the condition under which each formulation outperforms the other
- Can be used by smart preprocess/compiler based on different parameters
- Small expression of communication overload $\neq$ best algorithm
  - Berntsen's algorithm with the least communication overhead is the least scalable with $O(p^2)$ isoefficiency