Application with parallel programming of Strassen Algorithm

Final Project Proposal

The participants: 謝祥志、陳正宗

I. Introduction

Applications of matrices are found in most scientific fields. In every branch of physics, including classical mechanics, optics, electromagnetism, quantum mechanics, and quantum electrodynamics, they are used to study physical phenomena, such as the motion of rigid bodies. In computer graphics, they are used to manipulate 3D models and project them onto a 2-dimensional screen. In probability theory and statistics, stochastic matrices are used to describe sets of probabilities; for instance, they are used within the PageRank algorithm that ranks the pages in a Google search. Matrix calculus generalizes classical analytical notions such as derivatives and exponentials to higher dimensions. Matrices are used in economics to describe systems of economic relationships.

Volker Strassen first published this algorithm in 1969 and proved that the O (n^3) general matrix multiplication algorithm wasn't optimal. The Strassen algorithm O (n^2.807) is only slightly better than that, but its publication resulted in much more research about matrix multiplication that led to faster approaches.

II. Statement of problem

Matrix multiplication is often used when performing 3D transformations. In the application, the matrix multiplication definition algorithm is used to calculate it. This algorithm uses a lot of loop and multiplication operations, which makes the algorithm inefficient. The computational efficiency of matrix multiplication greatly affects the speed of the entire program, so some improvement of parallelization of the matrix multiplication algorithm is necessary.

III. Proposed approaches

The Strassen algorithm defines instead new matrices, only using 7 multiplications (one for each T_k) instead of 8. We may now express the $M_{i,j}$ in terms of T_k:

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

$$M_{11} = T_1 + T_2 + T_4 - T_5$$
$$M_{12} = T_5 + T_7$$
$$M_{21} = T_4 + T_6$$
$$M_{22} = T_1 + T_3 + T_6 - T_7$$

$$T_1 = (A + D)(E + H)$$
$$T_2 = (B - D)(G + H)$$
$$T_3 = (C - A)(E + F)$$
$$T_4 = D(G - E)$$
$$T_5 = (A + B)H$$
$$T_6 = (C + D)E$$
$$T_7 = A(F - H)$$

We iterate this division process n times (recursively) until the submatrices degenerate into numbers (elements of the ring R). The resulting will be padded with

zeroes just like A and B, and should be stripped of the corresponding rows and columns.

IV. Language selection

We choose Pthread , Cuda and OpenCL as our language. In order to make detailed configuration of computer architecture, we need Pthread as our tool language. Furthermore, the potential of heterogeneous computing can't be neglected, thus we choose Cuda and OpenCL to exploit the GPU power.

V. Statement of expected result

Because Matrix is the bottleneck of the Large Matrix calculation system, we hope to speed up by utilizing the concept of parallelization. In addition, we will make use of different toolkits to do the parallelization, and figure out which one suits our model the most.

VI. Timetable

Date	Plan to do
	Implement by Pthread/OMP.
	Implement by Cuda.
	Finish
	Prepare of the presentation
	Present

VII. Reference

1. https://en.wikipedia.org/wiki/Strassen_algorithm
2. https://en.wikipedia.org/wiki/Matrix_(mathematics)