Project CUDA : Solving a tridiagonal system on GPUs

Jovana Krstevska & Prédive Gopinathan

1 - **Problem**

- Finding a fast way to solve big tridiagonal systems on GPUs

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ \vdots \\ d_n \end{pmatrix}$$

2 - **Application**

- Various applications are possible, for example solving differential equations, where we usually find tridiagonal systems.

$$\begin{cases} -u''(x) + c(x)u(x) = f(x), \quad x \in (0,1), \\ u(0) = 0, \\ u(1) = 0. \end{cases}$$

$$A_h U_h = F_h$$

$$A_h = \frac{1}{h^2} \begin{pmatrix} 2 + c(x_1)h^2 & -1 & 0 & \dots & 0 \\ -1 & 2 + c(x_2)h^2 & -1 & \dots & 0 \\ 0 & -1 & 2 + c(x_3)h^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & -1 \\ 0 & 0 & 0 & -1 & 2 + c(x_n)h^2 \end{pmatrix}$$

$$U_h = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} \text{ et } F_h = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

3 – **Thomas algorithm**

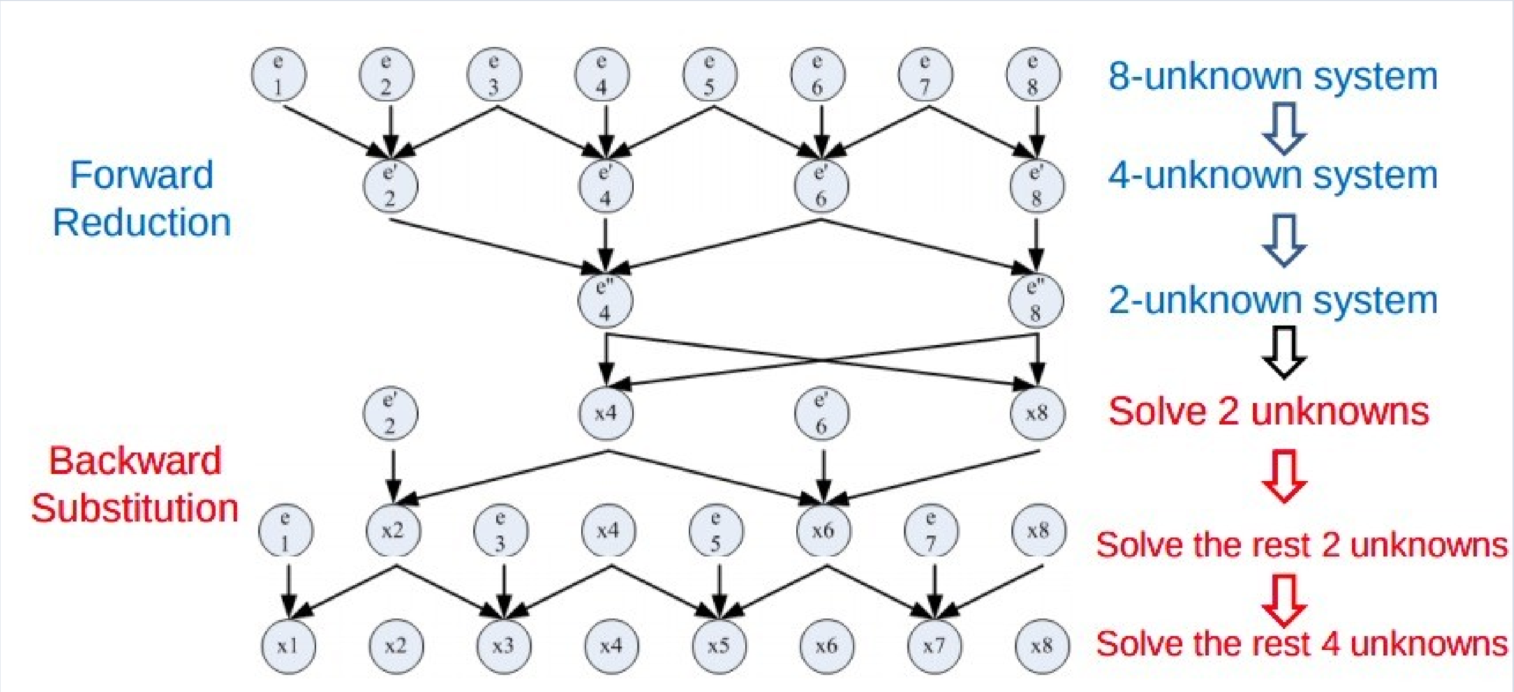
(gaussian elimination)

- For a system of N unknowns, the Thomas algorithm cannot be parallelized, and there are 2*N steps and 8*N operations

$$\begin{pmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & 0 & 1 & c'_3 & \\ & & 0 & 1 & c'_4 \\ & & & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} d'_1 \\ d'_2 \\ d'_3 \\ d'_4 \\ d'_5 \end{pmatrix}$$

4 – Cyclic reduction (CR)

- For a system of N unknowns, the CR algorithm uses N/2 threads, and there are $2*\log_2 N - 1$ steps and $17*N$ operations



5 – **Parallel Cyclic Reduction (PCR)**

- For a system of N unknowns, our PCR algorithm uses N/2 threads, and there are $\log_2 N$ steps and $12 * N * \log_2 N$ operations