CS315b : Sparse Conjugate Gradient in Regent

Léopold Cambier

ICME, Stanford University

December 6, 2016
The Problem

\[ Ax = b \]

where

\[ A \succ 0 \]

(i.e. \( x^\top A x > 0, \forall x \neq 0 \)) and additionally,

\[ A \text{ is sparse} \]

(i.e. \( \#\text{nnz}(A) \ll O(n^2) \))
Why do we care?

\[ \nabla \cdot (a \nabla u) = f \]
Why do we care?

\[ \nabla \cdot (a \nabla u) = f \]

\[ (\nabla^2 + k^2)u = f \]
Why do we care?

\[ \nabla \cdot (a \nabla u) = f \]

\[ (\nabla^2 + k^2)u = f \]

\[ \min_x \frac{1}{2} x^\top A x - b^\top x \]
Why do we care?

\[ \nabla \cdot (a \nabla u) = f \]

\[ (\nabla^2 + k^2)u = f \]

\[ \min_x \frac{1}{2} x^\top A x - b^\top x \]

\[ \min_x f(x) \]
Conjugate Gradient Algorithm

Iterative method
Conjugate Gradient Algorithm

Iterative method

- $r_0 = b,$
- $p_0 = b,$
- $x_0 = 0,$
- While $r_k^T r_k > \epsilon,$
  - $\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$
  - $x_{k+1} = x_k + \alpha_k p_k$
  - $r_{k+1} = r_k - \alpha_k A p_k$
  - $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
  - $p_{k+1} = r_{k+1} + \beta_k p_k$
Conjugate Gradient Algorithm

Iterative method

- \( r_0 = b \),
- \( p_0 = b \),
- \( x_0 = 0 \),
- While \( r_k^\top r_k > \epsilon \),
  - \( \alpha_k = \frac{r_k^\top r_k}{p_k^\top A p_k} \)
  - \( x_{k+1} = x_k + \alpha_k p_k \)
  - \( r_{k+1} = r_k - \alpha_k A p_k \)
  - \( \beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k} \)
  - \( p_{k+1} = r_{k+1} + \beta_k p_k \)
What and How to Parallelize?

- Bottleneck is clearly the sparse product

\[ Ap_k \Rightarrow O(k) \]

- Rest is

\[ r_k^T r_k \Rightarrow O(n) \]

update \( x_k, r_k, p_k \Rightarrow O(n) \)
Partitioning the Data

We have

\[(Ap)_j = \sum_{i: A_{ji} \neq 0} A_{ji} p_i\]

So we build

\[G = (V, E)\]

\[V = \{i | 1 \leq i \leq n\}\]

\[E = \{(i, j) | A_{ij} \neq 0\}\]
Partitioning the Data: Destination Nodes
Partitioning the Data : Edges
Partitioning the Data : Source Nodes
Partitioning the Data

- Good thing: no reduction needed
- Issue: need similar number of edges in each partition (ok for physical systems usually)
Control Flow

- Initialization
- For $k = 0, 1, \ldots$
  - For $c = 1, \ldots, p$ task $(Ap)[c]$ (reads edges, reads/writes nodes)
  - For $c = 1, \ldots, p$ task $(p^\top Ap)[c]$ (reads nodes)
  - For $c = 1, \ldots, p$ task $(\text{update } x \text{ and } r)[c]$ (reads/writes nodes)
  - For $c = 1, \ldots, p$ task $(r^\top r)[c]$ (reads nodes)
  - For $c = 1, \ldots, p$ task $(\text{update } p)[c]$ (reads/writes nodes)
Benchmark 1

Janna@Flan_1565. 1564794 nodes, 57920625 edges.
Benchmark 1

1,564,794 nodes, 114,165,372 edges, spd.
On one node. 0.21 [s/it]
(Sequential code : 0.25 [s/it] on 1 cpu)
Benchmark 1

On two nodes  0.13 [s/it]
Benchmark 2

Janna@Queen_4147. 4147110 nodes, 162676087 edges.
Benchmark 2

4,147,110 nodes, 316,548,962 edges, spd.
Benchmark 2

On one node 0.41 [s/it]
Future work

- Memory issues prevent to go higher in the number of nodes → need to work on that
- Go higher in number of nodes and get weak/strong scaling curves
- Parallelize initialization
Questions?