Dynamic Programming on CUDA
Finding the most similar DNA

Marek Kiszkis
Krzysztof Zarzycki
Grzegorz Kokosiński

May 24, 2012
The kidney donor story

- We have a database of donees waiting for a Kidney transplant
  - We store their DNA, particularly 6th chromosome

- A donor appears
  - Which donee is the best candidate for a transplant? That has a high chance of accepting the organ?
  - The one with most similar DNA/6th chromosome

- We need to compare a donor’s DNA with donees’
  - Need to define similarity – global alignment score
    - Needleman-Wunsch
    - Other possibilities
      - Edit distance
      - Local alignment score (Smith-Waterman)

- To keep the kidney in a good condition we need to find the donee fast!
Problem definition

- We have a database of nucleotide sequences representing donees
- We have donor’s sequence
- We are interested in finding the sequence with the highest global alignment score
  - Alignment itself is not needed
  - All scores except the best one are not needed
- The length of 6th chromosome is higher than 170 M bps*
  - But we can limit it to relevant group of genes, total tens of millions bps
- Number of analyzed donees in hundreds
- The performance is of grave importance
One to one matching

- Needleman-Wunsch algorithm - Basic approach
  - Dynamic programming

- Formula

  \[ F(i,j) = G(F(i-1, j-1), F(i, j-1), F(i-1, j), c(i, j)) \]

  - Builds the matrix (F) of alignment scores
  - requires \( O(n^2) \) memory space
    - unacceptable
One to one matching

- Processing
  - Cells on an antidiagonal (orange cells) can be computed parallel – in one iteration
  - Iterative computation of next cells
    - Uses values from previous iteration (blue cells), the rest can be dropped
One to one matching

| A | C | T | T | A | C | G | T | A | A | A | C | G | T | C | G | T | A |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| C |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| G |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| T |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
One to one matching
# One to one matching

|   | A | C | T | T | A | C | G | T | A | A | A | C | G | T | C | G | T | A |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| C |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| G |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| T |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| A |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
One to one matching
One to one matching

- The horizon
  - line of cells connecting bottom-left to upper-right corner
  - Represents the current state of computation
  - Computations push horizon forward
  - Linear memory
  - In-place processing
  - Alignment is lost
One to one matching

- Divide the matrix into parts
  - Blocks of rhombus shape

- Two levels of parallelism
  - Intra-block
  - Inter-block
One to one matching

- **Intra-block parallelism**
  - Working on Horizon fragment that fills shared memory
    - All cells from that fragment are updated in one step
One to one matching

- **Inter-block parallelism**
  - Dependencies between blocks
    - All blocks which have their dependency blocks processed, can be processed in parallel
  - Horizon moves while we process blocks
  - One block processed by one SM
Pruning technique

- The optimistic score for cell
  - For each cell we can compute the best (most optimistic) score of an alignment which contains this cell
    - Assuming the most optimistic scenario:
      - The rest of sequences match
      - There is one continuous gap
Pruning technique

- **Assume that we have some previous score P**
  - Pruning threshold
  - We are looking only for better score

- **Cell pruning**
  - Having the optimistic score O for a cell
    - If O < P
      we don’t need to process this cell
    - Otherwise, we process it normally
Pruning technique

- Cell pruning propagation
  - If all dependency cells from previous iterations are pruned, dependant cell can be pruned as well

\[
F(i,j) = G(F(i-1, j-1), F(i, j-1), F(i-1, j), c(i, j))
\]
Pruning technique

- **Block pruning propagation**
  - If all cells of initial horizon for given block are pruned, whole block can be pruned

![Diagram showing block pruning propagation](image-url)
Pruning technique

- **Block pruning propagation**
  - If all dependency blocks are pruned, dependant block can also be pruned
Pruning technique

- Block pruning propagation
  - It follows that more blocks can be pruned
Pruning technique

- **Trial run**
  - Search space is artificially limited to few %
  - Find some non-optimal score

- **Full run**
  - Uses the previously found non-optimal score to prune the search space, to find only the better solution
One to many

- Parallel comparison of each sequence with the given pattern
  - Sequences are distributed over nodes in a cluster
  - We looking only for the score of the best match
  - One global pruning threshold
    - Can be updated at any time
    - Used by all nodes for pruning in their computations
Results and conclusion

- The horizon - we optimized the memory usage
  - We can process very long sequences

- Without pruning, we achieve 20 GCUPS
  - On one C2050 GPU

- With pruning, if sequences are very similar
  - We drastically limit the search space of solutions – down to 5%
  - It makes matching up to 20x faster
  - The more similar the sequences are, the more computation can be pruned

- Very scalable and parallelizable in clusters