Fine-Grained Parallel Genomic Computation

Dominique Lavenier
CNRS / IRISA
Rennes, France

GenBank
*nucleotide database*

\( \times 1.65 \text{ / year} \)
Microprocessor Performances

Data vs proc. performances

2006 : GenBank 120 Gbp
  computational power: 12 Gbp/sec
  \rightarrow time: 10s

2010 : GenBank 900 Gbp
  computational power: 40 Gbp/sec
  \rightarrow time: 25s

2014 : Genbank 6.6 Tbp
  computational power: 130 Gbp/sec
  \rightarrow time: 50s
Genomic processing

- High potential parallelism
  - Very efficient on large clusters
    - Data are dispatched on the nodes
    - Processes run independently
    - Low overhead merging

Fine-grained processing

- FPGA processor
- Reconfigurable hardware
Content

- Molecular biology
- Genomic processing examples
- Algorithms
  - Dynamic programming
  - Seed-based
- Hardware
DNA → Protein

Genome

Gene

Protein

Genetic code

Nucleotide (nt) Base pair (bp)

ATG CCA GGG ATT AGG AGA TGG ACC TAA

Stop

MP GIR R WT

Folding

Amino acid (aa)

Gene : text / function relationship

Gene A → Protein A → Function

30% identity → same 3D structure

Gene X → Protein A’
Molecular biology
Genomic processing examples
- Shotgun sequencing
- Content-based search
- Motif detection

Algorithms
Hardware

Example 1
Shotgun sequencing

Genome
Duplicated
Random cut into small fragments (1000 bp)
Fragments sequencing
Assembling

attgaccagattgac
tggaccatatgg
ccatagagcacgac
tgagcacaggat
ggataggaccac
Example 1

Shotgun sequencing

Genome duplication

Random cut into small fragments (1000 bp)

Fragments sequencing

Assembling

Human

- 3 x 10^9 genome size
- 10 X coverage
- 3 x 10^7 number of fragments
- 5 x 10^15 pairwise comparisons

Pairwise comparison

Substitution error

Insertion/deletion error

Errors:
- Sequencing machines are not perfect
- Polymorphism (when a few individuals are sequenced together)
**Example 2**

**Content-based search**

unknown gene?

find in the genomic databases genes having similar text

(input)

(query sequence)

 databases

((~10^3 - 10^4 bp))

((~10^10 - 10^11 bp))

(List of alignments)

---

**Alignments**

Sequence #1

```
ATTAGAGCACTATAGAGGACACGATATAGAGCACACGAGTTAGGACCAGGGATTAGGGAGAGGACTAAATATAGAGCATATAGGAGGTAGGATCCAGGGATTAG
```

Sequence #2

```
TGGACCAGATTAGAGGACACGATATAGAGGATATAGGAAGGTAGGACCGGTAGACCAGTTAGACCAGGGATTTAGGGGGGATTTAGGAGTTAAA
```

Finding a local alignment we don't know:
- its length
- its position in both sequence
### Alignment scoring

Score = $21 \times 1 - 2 \times 3 - 2 \times 2$

= 11

Necessary to compare alignments $\rightarrow$ extract the best ones

---

### Content-based search

- **Activity #1**
  - May represent more than 70% of bioinformatics computation

- **NCBI (National Center for Bioinformatics)**
  - Handle more than 100,000 queries each day

- **Have to deal with**
  - The exponential growth of the databases
  - The increasing user needs
Example 3

Motif detection

- Gene families may contain specific signatures
- A signature is a pattern of amino acids (aa)
- Example:
  - $M \times(1-3) \ K \ P \ L \ [I \ V \ L] \ \times(2-4) \ [R \ S] \ L$
  - Can be expressed by regular expressions

Complex motifs

- 2D structure
- Can be expressed by string variable grammars
- Example
  - ATGCTGCTGATAGCATAACTGCTGCTGAAC

bio-palindrome
repeat

Y:3 $Y \times(2-6) \ X:4 \times(3-6) \ \bar{X} \times(4) \ AA$
Motif search

find in the genomic databases genes having this signature

List of genes

input

MOTIF genome (~10⁹ bp)

Motif search

signature

Motif detection (errors allowed)

a few days of computation

Olfactory gene family
600 known genes (2004)

Dog olfactory gene discovery

Sequencing 7.5 x 2 (2004)

36 x 10⁶ fragments

65 000 fragments

targeted assembly

1100 genes (500 new)

5 specific protein motifs

Motif discovery

DNA

Motif discovery
Genomic Processing features

- **Data**
  - Sequences
    - DNA (4 letters) – protein (20 letters)
  - Size: 10-100 Gbp (1 Tbp → 2010)

- **Algorithms**
  - Mostly string processing
    - approximate sequence comparison
  - Integer operation
  - Reduced arithmetic (8 – 16 bits)

---

Content

- **Molecular Biology**
- **Genomic processing examples**
- **Algorithms**
  - Dynamic Programming
  - Seed-based
- **Hardware**
Dynamic programming

- Input: 2 (text) sequences
- Output: 1 alignment
- Idea:
  - Find the minimal number of operations to move from sequence #1 to sequence #2
- Needleman & Wunsh (1970)
  - Global alignment
- Smith & Waterman (1981)
  - Local alignment

Needleman & Wunsh – Global alignment

Sequence #1 (size N)

Sequence #2 (size N)

Complexity: $N^2$

$N = \text{sequence length}$
Dynamic programming

Smith & Waterman – Local alignment

Sequence #1 (size N)

Sequence #2 (size N)

Complexity : $N^2$

$(N = \text{sequence length})$

$F_{\text{Local}}(M_{i,j-1}, M_{i-1,j}, M_{i,j})$

Best score

Sequence #1 (size N)

<table>
<thead>
<tr>
<th>A</th>
<th>T</th>
<th>A</th>
<th>G</th>
<th>A</th>
<th>C</th>
<th>T</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sequence #2 (size N)

<table>
<thead>
<tr>
<th>A</th>
<th>T</th>
<th>C</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A T A G A C T G

A T C G A . T G

Data dependencies

Sequence #1

A T A G A C T G

A T C G A . T G

Computation requires $2N - 1$ steps

Possible speed-up : $N^2/2N-1 = N/2$
**2D systolic architecture**

1 systolic cell performs:
\[ F(M_{i,j}, M_{i,j+1}, M_{i+1,j}) \]

- \( N^2 \) cells \( \Rightarrow \) speed-up \( \frac{N^2}{2N-1} \) \(~ N/2\)

- Bad efficiency

---

**1D systolic architecture**

2D array:
- only one diagonal active at each step

1D array:
- one cell emulates one column

- Step 3

- \( N \) cells \( \Rightarrow \) speed-up \(~ N/2\)

- Better efficiency than 2D structure
Database scanning

Speed-up = \frac{N \times M}{N \times M - 1} = \sim N \quad (N \ll M)

Dynamic Programming

Conclusion

- Advantages
  - Gives the exact solution
  - Efficient parallelization on 1D systolic architecture

- Drawback
  - Too high complexity (N^2) to handle large databases with sequential machines
    - Scan of GenBank can take a few hours

- Need of heuristic to decrease computation time
  - FASTA (Pearson, Lipman 1988)
  - BLAST (Altschul et al. 1990)
Seed-based algorithm idea

Local Alignments

Programming dynamic:
1. Full matrix computation
2. Highest scores detection
3. Trace-back

Seed-based idea:
1. find "high similarity" regions from seeds
2. limit the search space near these regions

Seeds?

- A common word of K characters includes in both sequence

ATTAGGACCATTAGGACGACCTCCAGGATAGGACCGAGGATAGGACCCAGGAGTTAGAC

TTTAGGACACGAGGATATGGACCAGGCCCTTAGGCTGGACCGTGGTTAG
Seed?

- A common word of K characters includes in both sequence

ATTAGGCCATTAGGACCGACCCTCCAGGATAGGACCCAGGAGTTAGAC

TTTAGGACAGGATATGGACCAGGCCCTTAGGCTGGACCGTGGTTAG

GACCATTAGGCCAGGACC . T

G . CCCCTAGGCGTTGACCCT

Seed based algorithm efficiency

- 100X – 1000X time faster than Programming Dynamic
  - Space search is drastically reduced
    ➔ default k-word size = 11
- Large databases can be scanned in a few seconds
- Is there still room for fine-grained parallelism?
Seed based algorithms

- k-word hit
- Ungapped alignment
  if score > threshold
- gapped alignment

Profiling
- ~ 85%
- ~ 12%

Parallelization of seed based algorithms

2 sub levels of parallelism
1. multiple gapped/ungapped alignments can be performed simultaneously
   - high data throughput
2. alignment computation
   - DP on a restricted area
Dynamic programming on a restricted area

1D systolic architecture

Search space around D diagonals → D cells

Seed-based algorithms

Conclusion

- Very powerful heuristic
  - BLAST software can compute hundred of Mbp/sec
- Adopted by the scientific community
  - BLAST is the reference software
- High potential for fine-grained parallelism
  - However: more complicated than dynamic programming (data access)
Content

- Molecular Biology
- Genomic processing examples
- Algorithms
  - Dynamic Programming
  - Seed-based
- Hardware

2 main classes of applications

- Intensive computation
  - Shotgun
  - Genome structure analysis
    - Ex: Repeat sequences
  - Genome comparison
    - Ex: Human Λ mouse

- Database querying
  - One (small) query against the databases

must be done as fast as possible

must be done in a few seconds
Clusters

- Coarse grained parallelism
- Well suited for intensive computation
  - Large tasks can run independently
  - Limited I/O operations: data are intensively re-used
- Database querying:
  - Data access can be a bottleneck

VLSI Accelerators

- A few realizations essentially for speeding up DP algorithm (systolic array)
- Advantage
  - Very fast: 50X – 100X
- Drawback
  - Chip need to be periodically redesigned
    - Expensive, niche market
  - Lack of flexibility
    - S&W algorithm
FPGA Accelerators

- Suppress the VLSI drawback
  - Standard components
  - Can be reconfigured indefinitely
- Commercial products available
  - Ex: Decypher Engine from TimeLogic Inc.
    - S&W algorithm
    - BLAST-like algorithm
    - HMM motif search
    - ...
- Performances: 50X – 100X

Hardware accelerators

General conclusion

- Fine-grained parallelization
- Speed-up: 50 to 100 compared to microprocessors
- Can be mixed with clusters
  - i.e. node = 1 processor + 1 accelerator
- Well suited for intensive computation
- Database querying
  - Do not solve the data access problem
Querying databases

- First bioinformatics activity
- Genomic databases grow faster than
  - processor performances
  - data access time (disks!)
- Solutions
  - Keep the data in main memory and parallelized the accesses
    → Databases are dispatched on memory cluster nodes
  - Design a large parallel fast access memory