



# Bioinformatics in Biomining

Nicolas Loira, LBMG  
University of Chile  
April 2006



# Contents

---

Bacteria

Algorithms

Machines



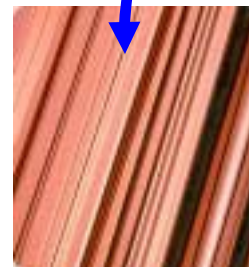
# Contents

---

- **Biomining**
- **Biotechnology**
  - Pattern Matching
  - shift-or
  - Itanium
  - Benchmark

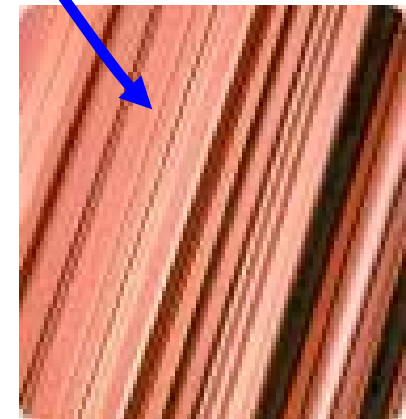


# Copper in Chile





# Biomining



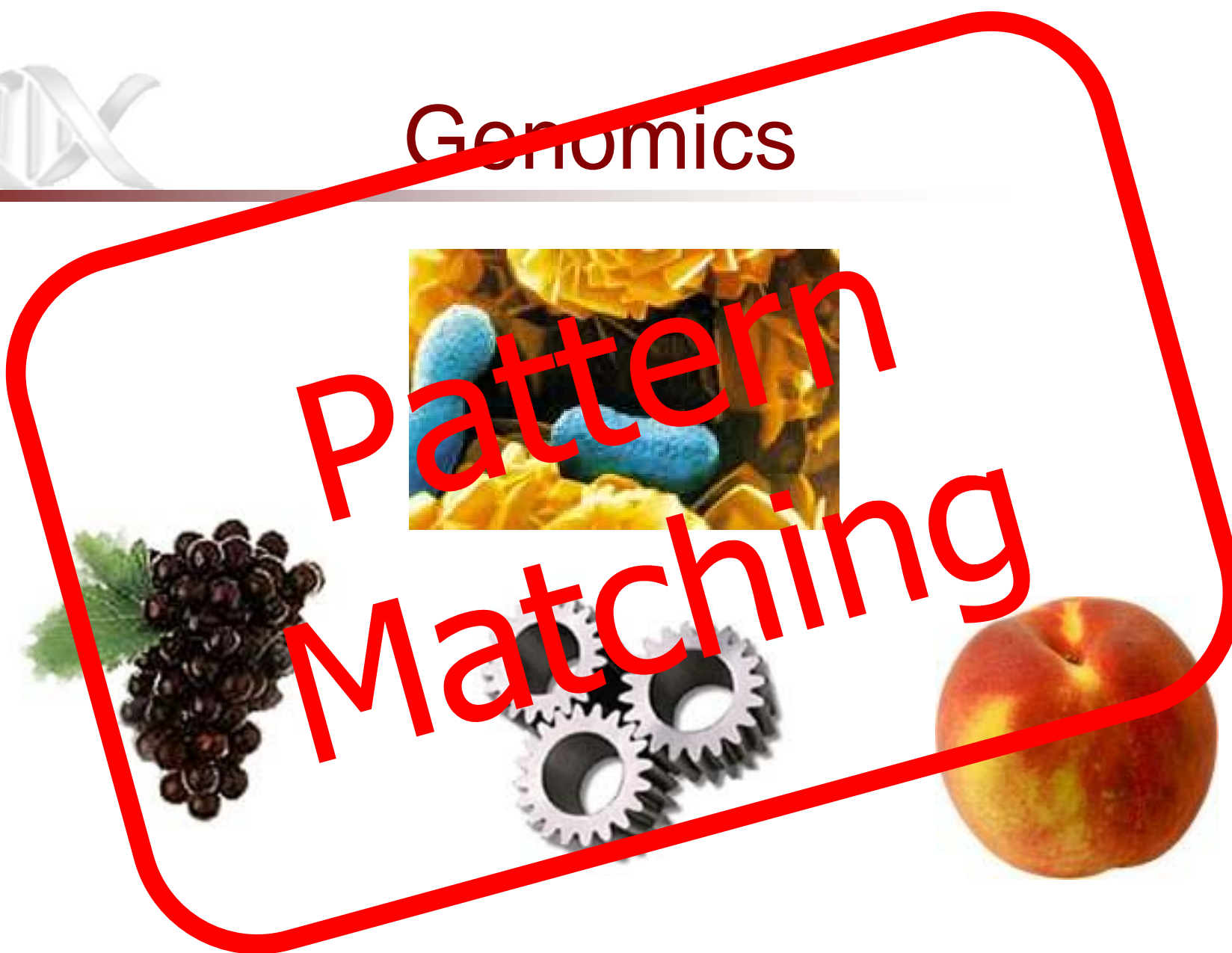


# Genomics





# Genomics





# Algorithms & Tools

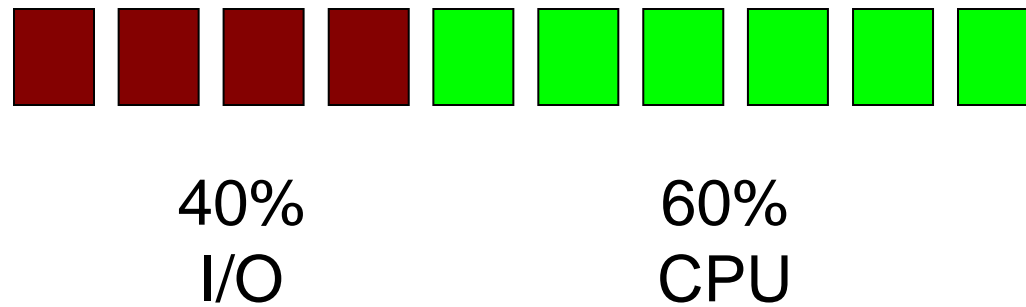
ATGCCTGA-----  
-----CTGCTGCC

- Smith-Waterman
- Needleman-Wunsch
- FastA
- BLAST
- Phrap
- Lots of ad-hoc variations

-2 -2 -2 -2 +1 +1 +1 -1 -2 -2 -2 -2

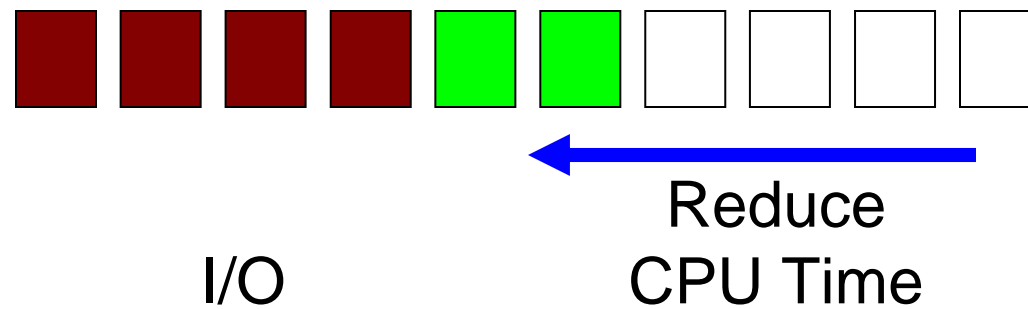


# Processing vs. I/O





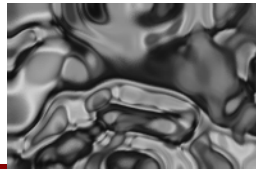
# Processing vs. I/O





# Identification of bacteria

- To detect the presence of a given organism we use a specific biomarker
  - Synthesized DNA sequence 20-60bp
  - Act as a fingerprint
  - Hybridizes with target genome
- Biomarkers can be of different specificity
  - Can detect genus, phylum, species, etc.





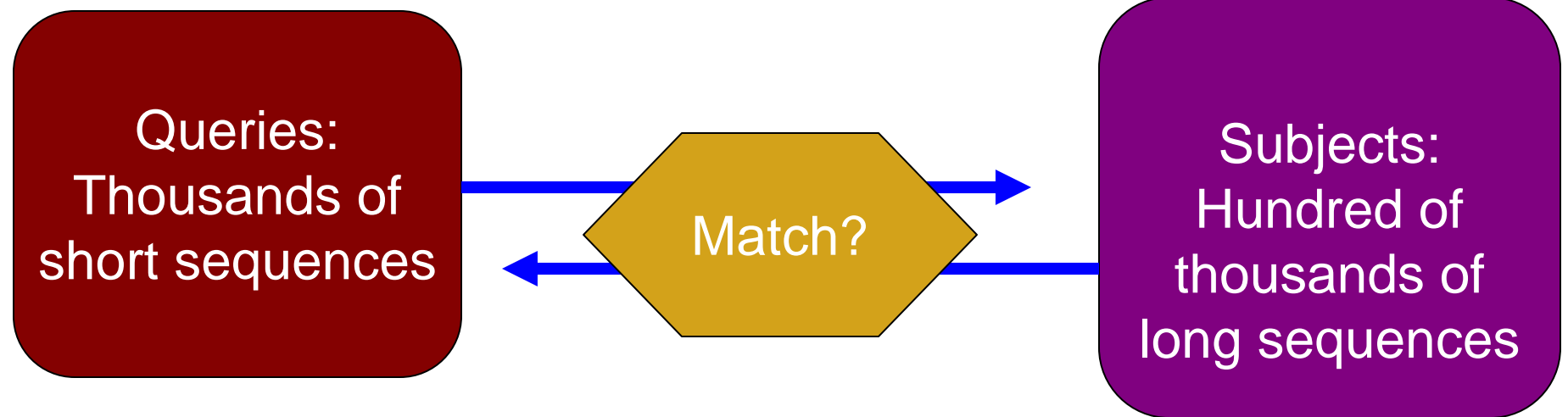
# Statement of the problem

- Given a set of candidate biomarkers (a few thousands)
- Given a big database of known genomes ( $\approx 10^5$  entries 1000bp long)
- Determine which biomarker occurs in which genome, allowing less than 3 substitutions

Where bp = base pairs  $\approx$  letters  $\approx$  bytes



# Searching biomarkers





# No indexing

- New query sequences each day
- New subject sequences frequently
- Indexing 20bp seqs with at most 3 mismatches means indexing all 5bp sequences
- Every subject will match
- Does not worth it!



# Why not use indexes?

- Searching for a pattern of length  $p$  accepting  $m$  mismatches corresponds to  $(m+1)$  exact searches of length  $p/(m+1)$
- We have to check  $(m+1)N(s-m)/4^{p/(m+1)}$  sequences.
- In our case  $m=3$ ,  $s \gg 1000$  and  $p=20$ , so each index has  $1000/4^5 N \gg N$  sequences.
- Also, using index we don't take advantage of CPU's cache.



# Algorithm?

- Shift-or: Baeza-Yates, Gonnet
- Exact match
- Variation to support h-mismatches
- Implementation is very fast!  
(how *fast* is fast?)
- Real runs: days and even weeks



# Shift-Or Query

query=ACTGATATCTGACTACTGTA

Q[A] = 01110101111011011110

Q[C] = 10111111011101101111

Q[G] = 11101111110111111011

Q[T] = 11011010101110110101



# Shift-Or Subject

S=...ACTGATCAGCTATCGAGACTAGC...

$$\text{Status} = (\text{Status} \ll 1) | Q[S[i]]$$

Status = 11111111111111111110

Status = 111111110111001110

Status = 111101111111111110



Win condition!



# New cluster of Itaniums!

- 32 Itanium2 (1.6 GHZ, 2Gb RAM)
- Infiniband
- But our problem (as typical in bioinformatics) is embarrassingly divisible
- So the challenge is to make each Itanium process sequences as fast as possible
- And discover how fast an Itanium can be!



# Implementations

- Pentium IV 3.6 GHz, 4 Gb Ram
- Itanium2 1.6 GHz, 2 Gb Ram
- C implementations for both platforms
- Compilers icc and gcc
- Simple assembler implementation for Itanium2 (based on icc version)

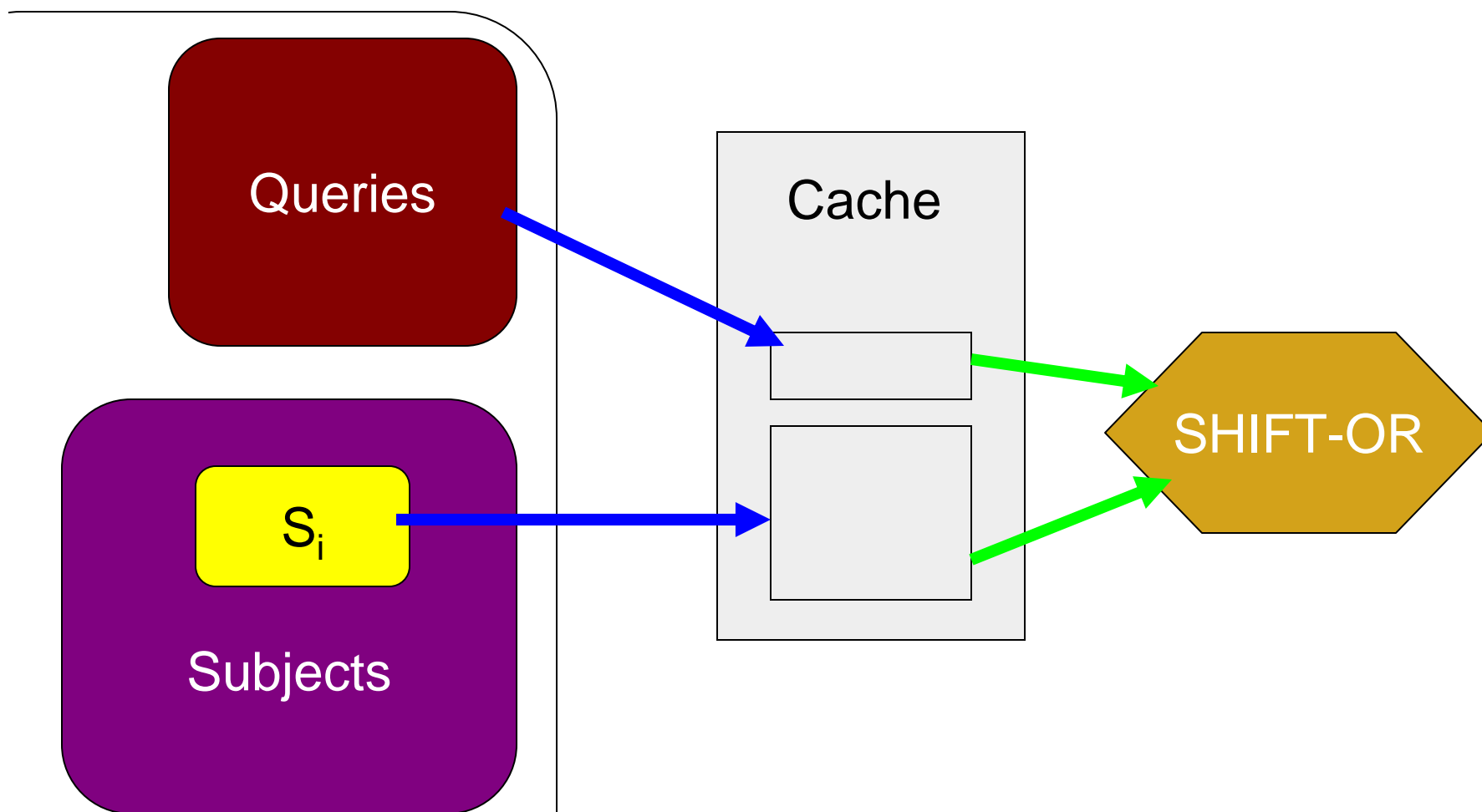


# Runners for fastest SOR

- PIV 3.6 GHz gcc v. 3.4.4
- PIV 3.6 GHz icc v. 8.1
- Itanium2 gcc v. 3.4.4
- Itanium2 icc v. 9.0
- Itanium2 asm v.0001

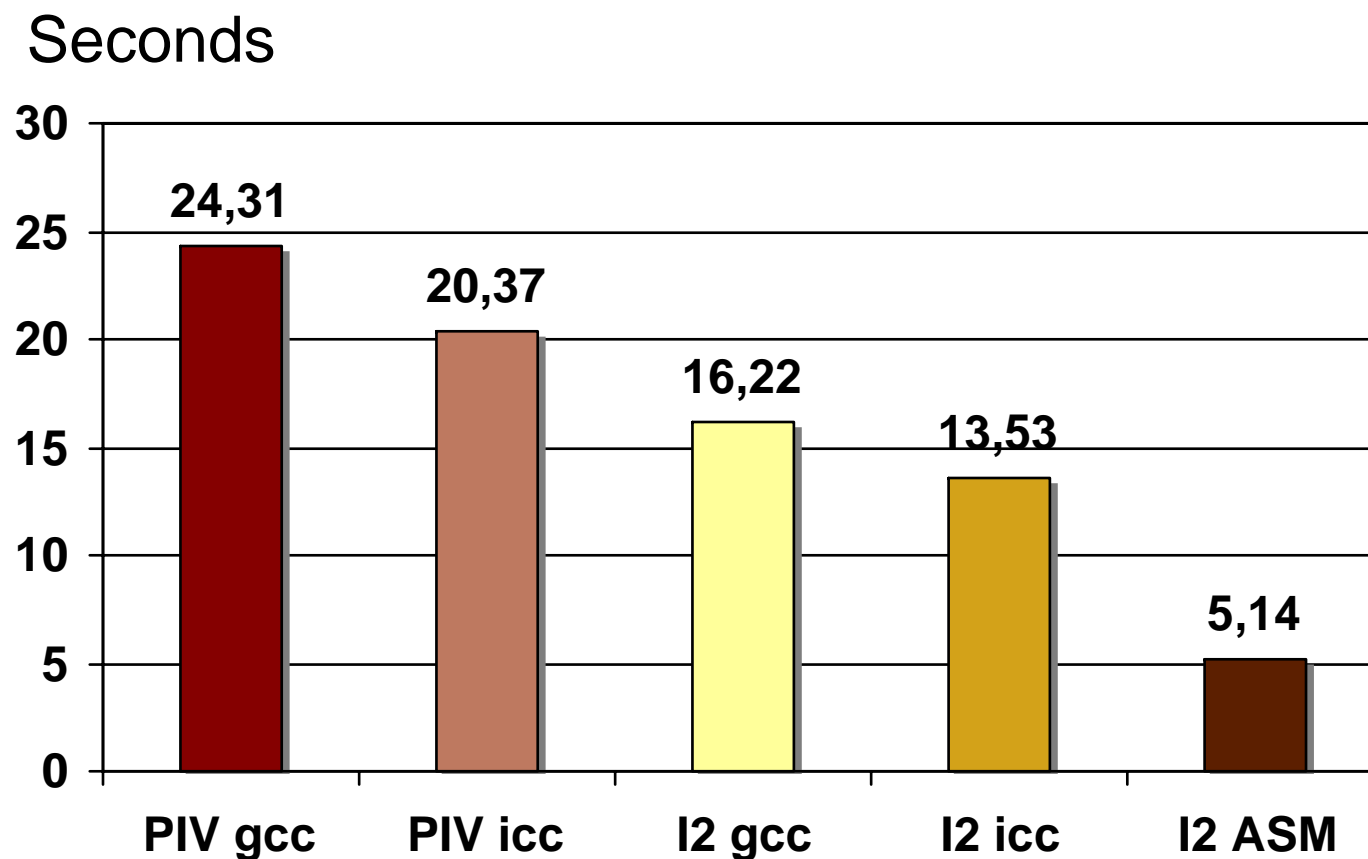


# Cache use





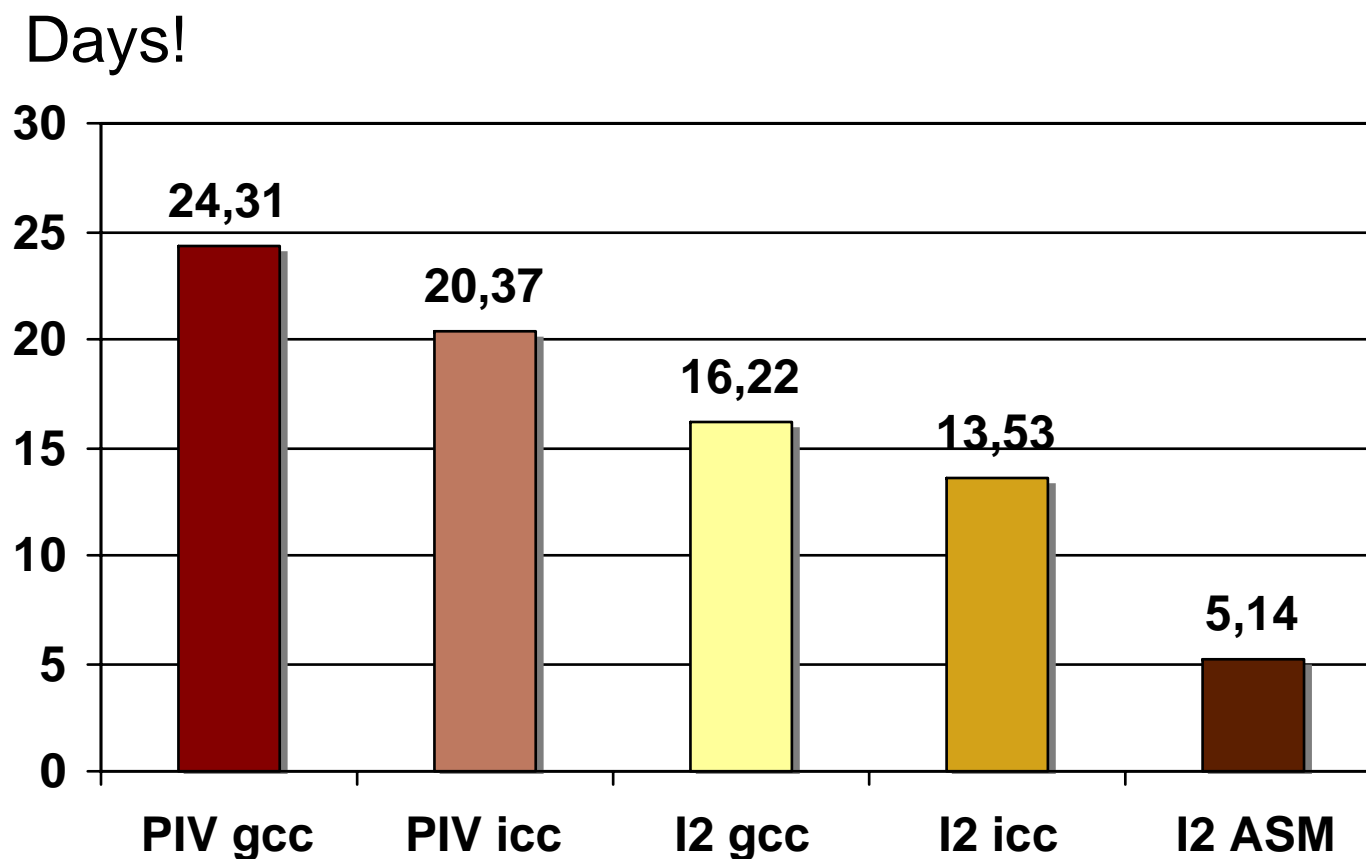
# Benchmark Results



Data set: 10 queries, 192.619 subjects



# Real Life Numbers



Data set: 90.000 queries, 192.619 subjects



# Assembler implementation

- Loop unrolling? -> software pipeline
- 64 bits registers ->
  - 8 letters at once
  - Patterns with length 64
  - Or two patterns of length 32



# Subtle optimizations

A = 65 = 01000001

C = 67 = 01000011

G = 71 = 01000111

T = 84 = 01010100



## Subtle optimizations

A = 65 = 0 1 0 0 0 0 0 0 1  
C = 67 = 0 1 0 0 0 0 0 1 1  
G = 71 = 0 1 0 0 0 0 1 1 1  
T = 84 = 0 1 0 1 0 1 0 0 0

Q[x]; r40+x; p16,17



## 64 bits register imply...

```
Query= [ACTGATA .. GACTACTGTA ]  
len(query) <= 64
```

```
Query = [ACTGATATCTGACTACTGTA ]  
Query1= [A T A A C G C A T T ]  
Query2= [ C G T T T A T C G A ]  
len(query1)=len(query2) <= 32
```

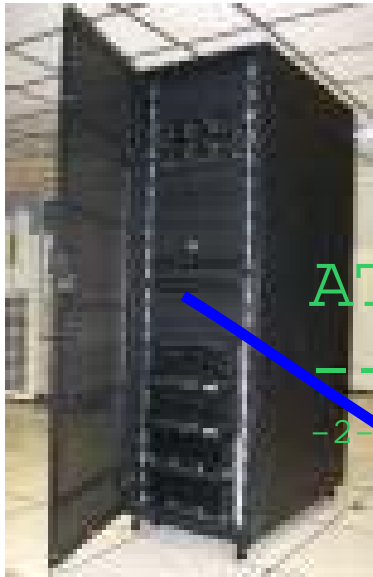


## Future steps

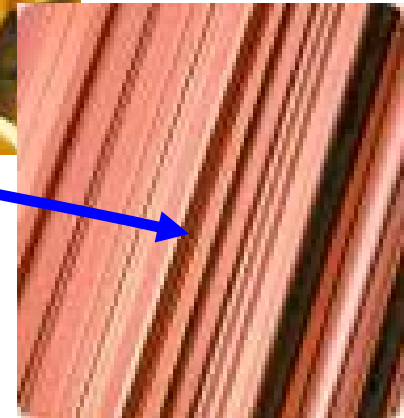
- Profiling, profiling, profiling...
- Pre-fetching and smart use of cache
- Sequence pre-formatting?
- Thread I/O
- Try new shifter in Montecito!



# Closing...



ATGCCTGA-----  
-----CTGCTG  
-2-2-2-2+1+1+1-1-2-





# Acknowledgments

- Andres Aravena
- Alejandro Jofré
- Alejandro Maass
- Servet Martinez
- Thanks to everyone at Gelato!



# Metagenomics

- Only 5% of bacteria have been identified
- Most bacteria can't be growth in lab conditions
- We take samples directly from ore (*in situ*)



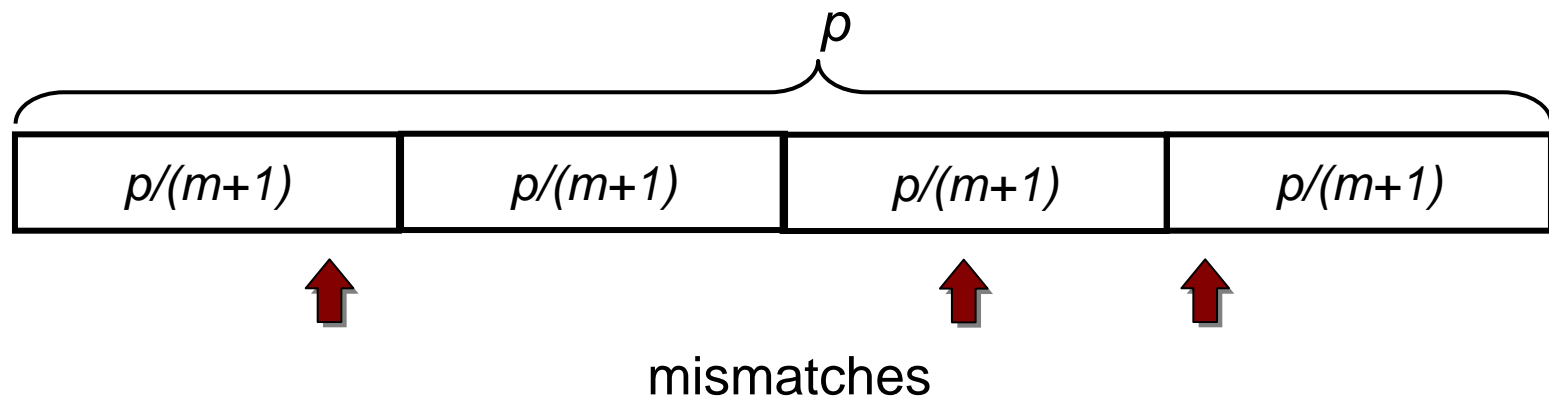
# Biomarker design

- Depending on the experiment objective, we need biomarkers binding to a given set of organisms and not binding in others
- Binding requires that most nucleotides coincide (but some substitutions can be allowed)



# About Indexing

- If a pattern of length  $p$  coincides with the text except for  $m$  mismatches, then at least one substring of size  $p/(m+1)$  is an exact match.





## ▪ About Indexing

- Searching for a pattern of length  $p$  accepting  $m$  mismatches corresponds to  $(m+1)$  exact searches of length  $p/(m+1)$
- Assuming a 4-letter alphabet (DNA), we split the search space in  $4^{p/(m+1)}$  exact indexes.



## Size of each index

- Each text sequence of average length  $s$  has  $s-m$  substrings of size  $(m+1)$ .
- If text database has  $N$  sequences, then there are  $N(s-m)$  occurrences of  $(m+1)$  substrings.
- Each index has  $N(s-m)/4^{p/(m+1)}$  entries on average.



## Using actual numbers

- So, using indexes, we have to check  $(m+1)N(s-m)/4^{p/(m+1)}$  sequences.
- In our case  $m=3$ ,  $s \approx 1000$  and  $p=20$ , so each index has  $1000/4^5 N \approx N$  sequences.
- Also, using index we don't take advantage of Itanium cache.

