Parallel large scale inference of protein domain families

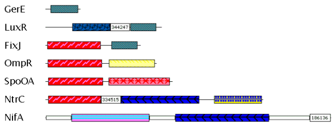
Daniel Kahn [2,3] Clément Rezvoy [1,3] Frédéric Vivien [2,3]

[1] ENS de Lyon [2] INRIA [3] Université de Lyon
France

Thursday June 3, 2010

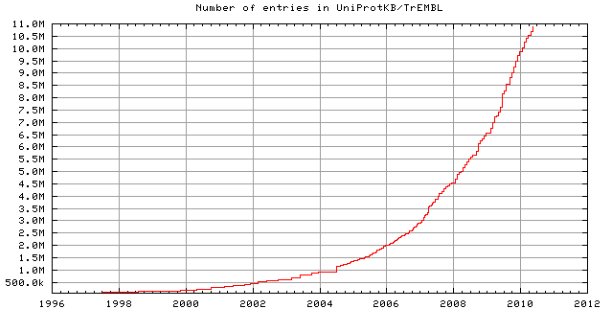
PRODOM: a repository of protein domain families

prodom.prabi.fr



- PRODOM is a widely used database of protein domain families
- PRODOM is automatically built from Uniprot database using MKDOM2
- MKDOM2 is a sequential algorithm of quadratic complexity

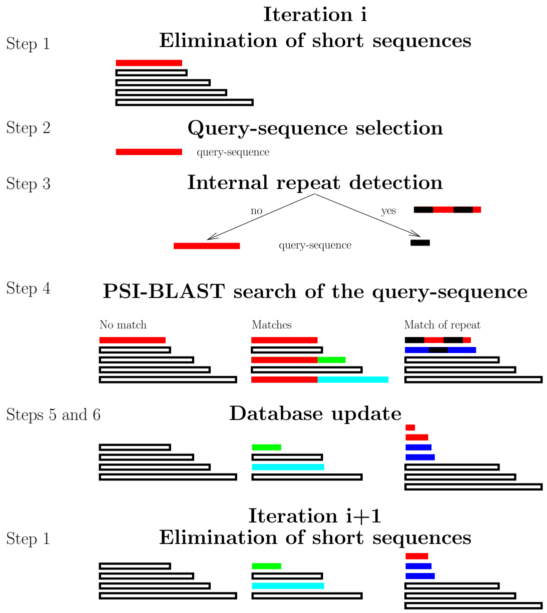
The problem



- Uniprot database size doubling roughly every other year
- 2002 version of ProDom: 2 months
- 2006 version of ProDom: 15 months
- 2010 version of ProDom: $>$ 10 years

The parallelization of Mkdom2 is mandatory to ensure ProDom's future

The MkDom2 algorithm



Approach 1: parallelizing each iteration

Pros

- Leave untouched the nature of the heuristic

Cons

- Each iteration is itself an iterative process
- Iterations are short:
  Medium test: 72,413 iterations averaging 0.86 seconds
- Strongly coupled fine grain parallelism

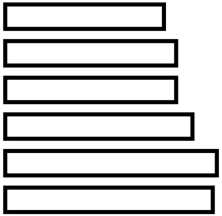
Approach 2: running iterations in parallel

Pros

- Enable to use thousands of cores in parallel

Cons

- Inter-dependences of iterations



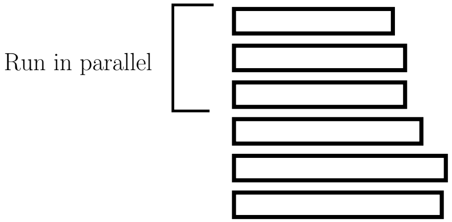
Approach 2: running iterations in parallel

Pros

- Enable to use thousands of cores in parallel

Cons

- Inter-dependences of iterations



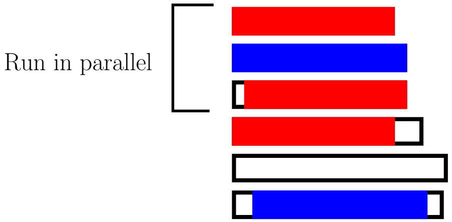
Approach 2: running iterations in parallel

Pros

- Enable to use thousands of cores in parallel

Cons

- Inter-dependences of iterations



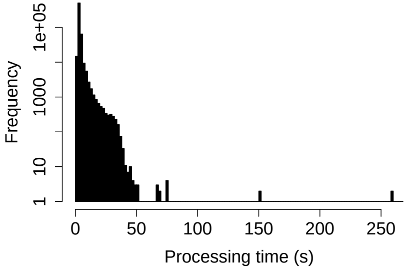
Approach 2: running iterations in parallel

Pros

- Enable to use thousands of cores in parallel

Cons

- Inter-dependences of iterations
- Variations in query-sequence processing times
  Median 2.5 s; Worst-case: 260 s



Approach 2: running iterations in parallel

Pros

- Enable to use thousands of cores in parallel

Cons

- Inter-dependences of iterations
- Variations in query-sequence processing times
- Impact on the underlying biological hypothesis

Prediction of dependences between query-sequences

Solution: All-against-all BLAST comparison

- BLAST far less stringent than PSI-BLAST
- Idea: do not simultaneously run queries from matching sequences
- May seem to double the overall computation time
- Can be trivially parallelized $\Rightarrow$ can lower wall-clock time

Naive Master-worker approach

```
While Database is not empty do
    1. Select the $n$ shortest and non-homologous sequences
    2. Send $\frac{n}{p}$ sequences to each of the $p$ workers
    3. Gather the $n$ results
    4. Update the database
```

Problems

- No overlap of the work of the master and of the workers
- Variations in query-sequence processing times
- Processor heterogeneity
- Potential communication bottlenecks
- Potentially high computational burden on the master

Problems and solutions

No overlap master/worker + unpredictable query processing times

- Master compute new batch of sequences when one worker has processed 50% of its last batch
- A worker receives a share only if it has started processing its previous batch

Processor heterogeneity

- Work distribution according to initial platform benchmark

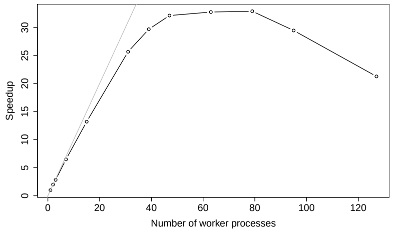
Master load and communication bottleneck

- Database update performed on each worker (not sent by master)
- All communications are asynchronous

Experimental setup

- MPI_MKDOM2
- Initial database (DB) of 556,964 sequences
- Recursively nested subsets DB/2, DB/4 and DB/8
- All-against-all blast search was run with a threshold of 10^{-4} on the E-value

Speedup



- Parallelization efficient up to 40 processes on DB/8
- The larger the database, the larger the achievable speedups
- going from 39 to 79 workers divided the processing time of DB/8 by 1.11 while that of DB was divided by 1.72

Conflict prevention efficiency

Number of workers	1	2	7	31	39	63	79	127
% of conflicts	0.37	0.47	0.70	1.35	1.59	3.13	3.75	5.11

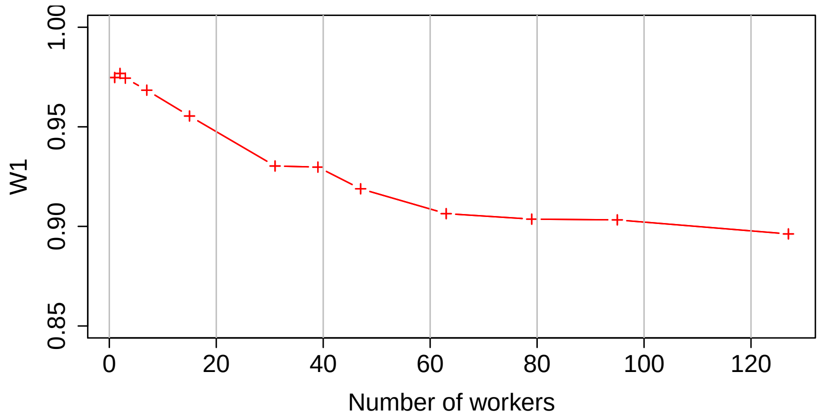
Table: Percentage of queries leading to conflicts as a function of the number of workers, for the processing of DB/8.

Conflict prevention efficiency

Number of workers	1	2	7	31	39	63	79	127
% of conflicts	0.37	0.47	0.70	1.35	1.59	3.13	3.75	5.11
% of conflicts w/o dependency information					25			

Table: Percentage of queries leading to conflicts as a function of the number of workers, for the processing of DB/8.

Result stability: comparing parallel and sequential results



- W1(fam_{seq}) : how well is fam_{seq} conserved in the parallel result.
- Good stability, even above maximum speedup.

Conclusion

- Able to handle huge unpredictable unbalance
- Rather good platform utilization
- Stability of the results *w.r.t.* the sequential algorithm
- New versions of ProDom built in the near future