Code performance analysis and improovement by optimasing compilation : Case study

Pierre Tassel under the supervision of Pr. Touati

Département of Computer Science
University of Nice Sophia-Antipolis

Student in Master of Computer Science

July 4, 2018

Outline

1. The program
2. Experimental approach
3. Analysis and optimization of sequential performances
4. Adding parallelism
5. Simple algorithmic modification
6. Conclusion
7. Perspectives

The program

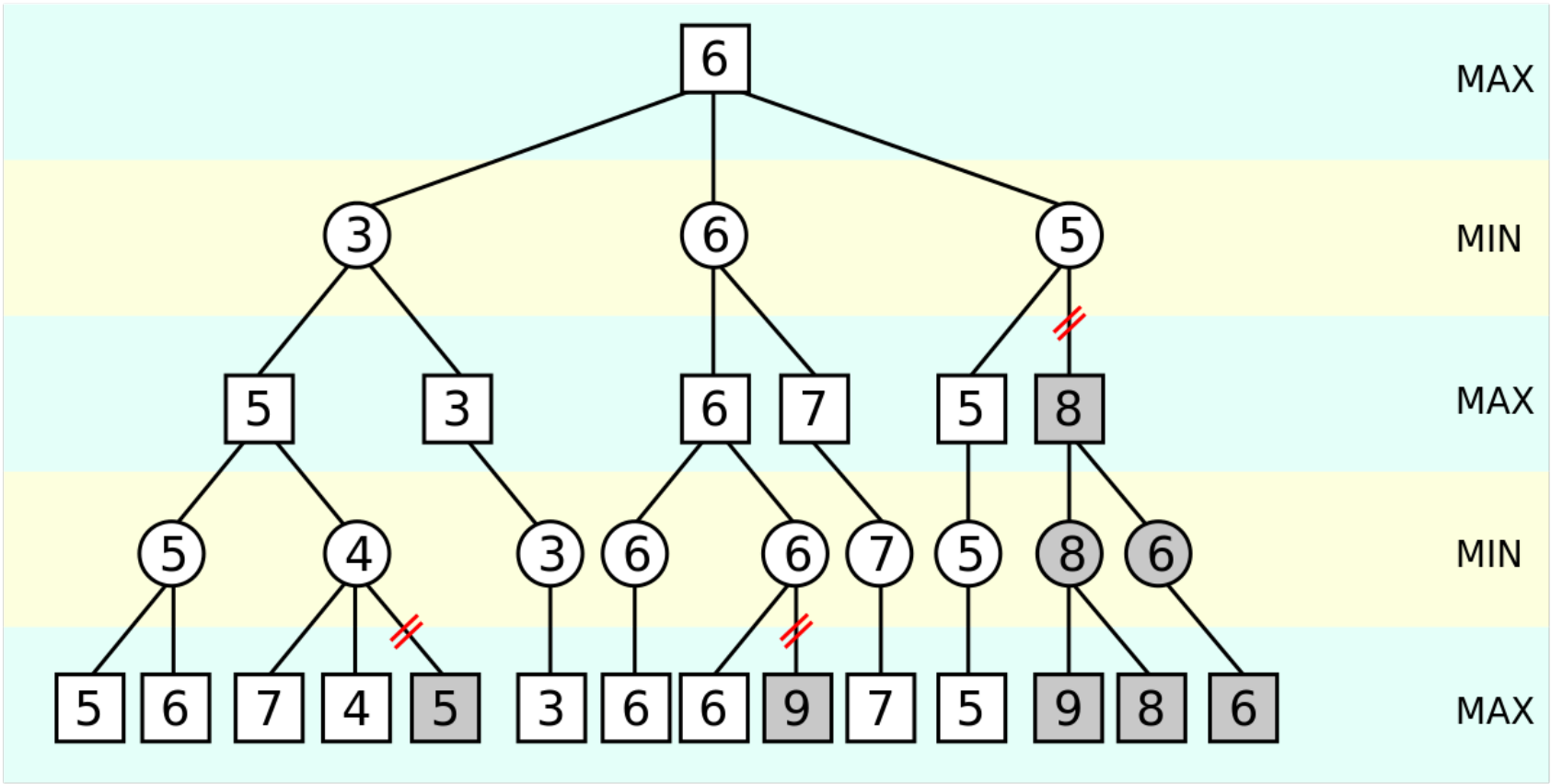
MinMax for the Awale game

- AI for Awale's game with 6 row and 4 seed per row.
- Writen in C++ by Pr Régin.

Figure: The Awale's game.

The program

MinMax with Alpha Beta prunning

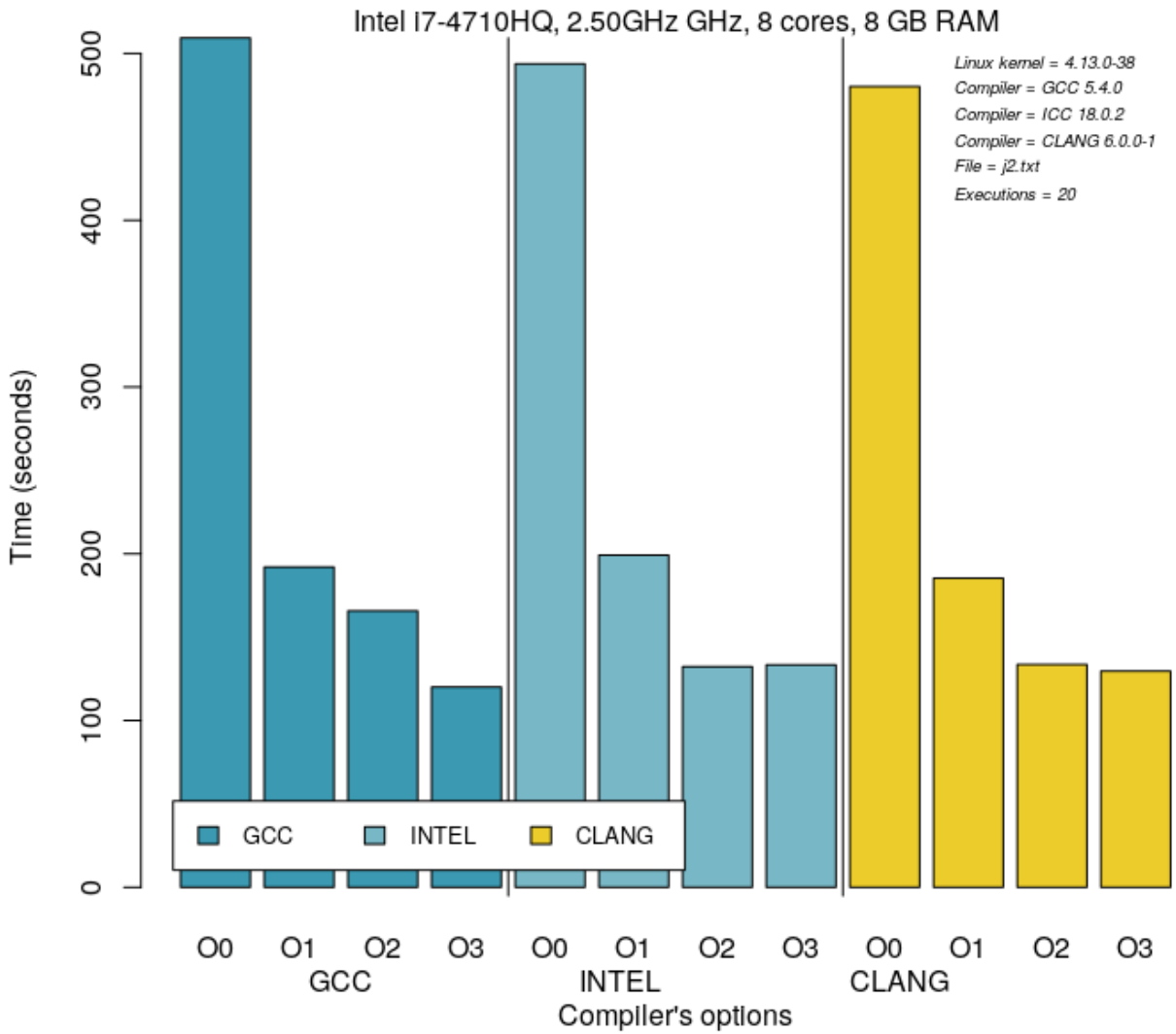


Experimental approach

Hardware and software environment

- *Processeur Scaling* desactivated, Intel core i7 8 core 2,5GHz processor, 8Go DDR3, 256Go of SSD.
- Minimalist CLI environement, Arch Linux based on Linux 4.14.13-1.
- Three compiler analysed : GCC, ICC and CLang (all up to date).
- Input file generated by playing the program against itself, deterministic program.
- The executions are repeated 20 times for each tested configuration.

Analysis and optimization of sequential performances



Code performance profiling

GProf

```
 %   cumulative          total
time   seconds   calls   s/call  name
48.83    167.72 5172852992  0.00  jouer_coup(Next*, Pos*, Pos*, int, int)
18.25    230.39 3262157329  0.00  copier(Pos*, Pos*)
17.22    289.56 3262157262  0.00  est_affame(Pos*, int)
 8.79    319.75 1457826958  0.00  calculer_coup(Next*, Pos*, int, int, int, int, bool)
 5.31    337.98 3252245939  0.00  valeur_minimaxAB(Next*, Pos*, int, int, int, int, bool)
 0.87    340.99 1775297034  0.00  evaluer(Pos*)
 0.52    342.77     67      0.03  test_fin(Pos*)
 0.09    343.65     33     10.34  decisionAB(Next*, Pos*, int, bool)
```

Figure: Code Profiling with GProf.

Code performance profiling

vTune

Elapsed Time: 118.666s

CPU Time:	**117.755s**
Instructions Retired:	588,377,500,000
CPI Rate:	0.499
CPU Frequency Ratio:	1.000
Total Thread Count:	1
Paused Time:	0s

Top Hotspots

This section lists the most active functions in your a

Function	Module	CPU Time
jouer_coup	advisor.out	70.132s
calculer_coup	advisor.out	15.441s
copier	advisor.out	14.165s
est_affame	advisor.out	7.176s
valeur_minimaxAB	advisor.out	5.755s
[Others]		5.087s

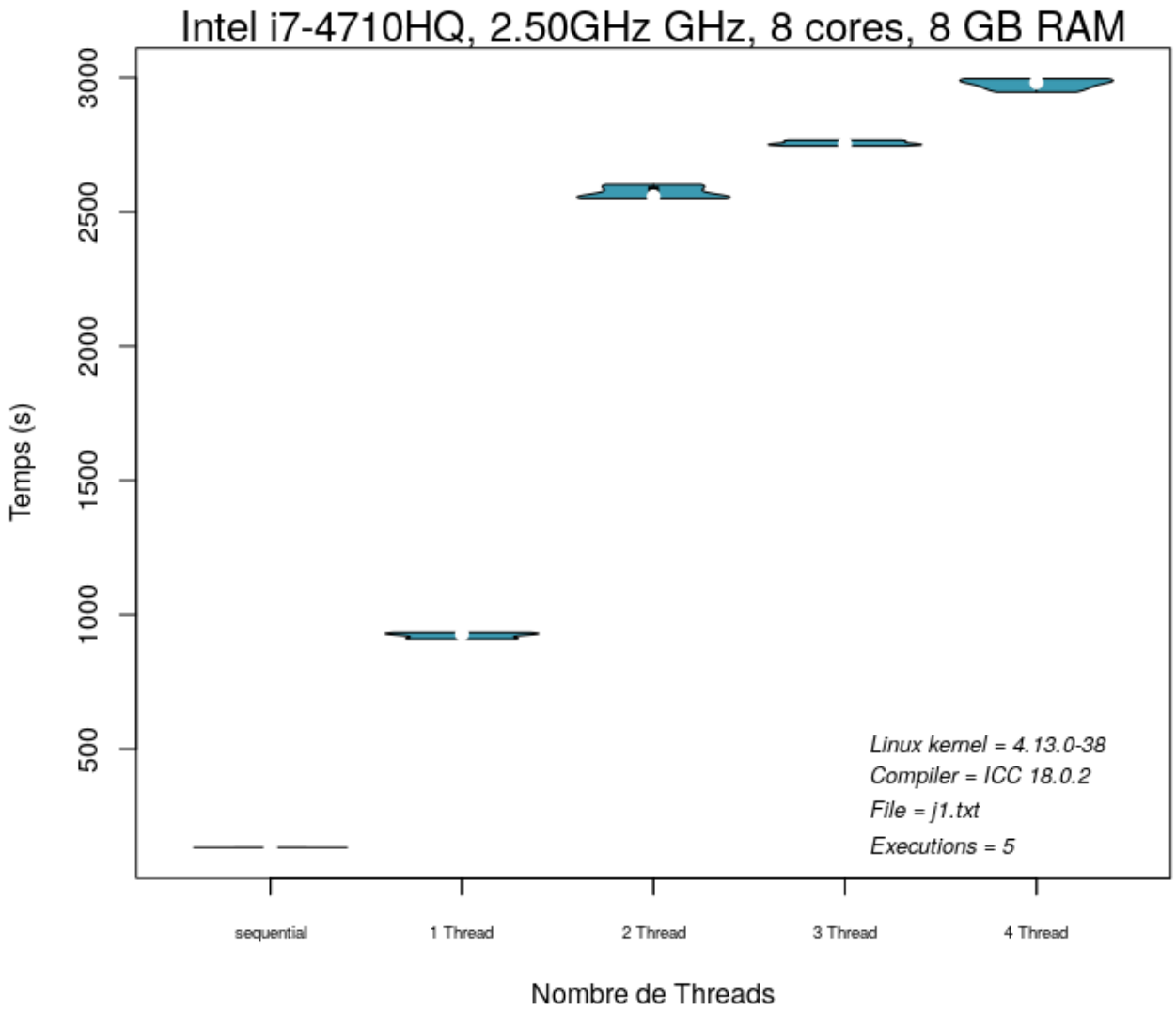
Figure: vTune processor usage analysis.

Elapsed Time: 120.174s

CPU Time:	118.454s
Memory Bound:	**2.7%**
L1 Bound:	6.6%
DRAM Bound:	
DRAM Bandwidth Bound:	0.0%
LLC Miss:	0.1%
Loads:	157,360,120,662
Stores:	100,867,825,944
LLC Miss Count:	900,054
Average Latency (cycles):	8
Total Thread Count:	1
Paused Time:	0s

Figure: vTune memory usage analysis.

Naïve approach to adding parallelism



False Sharing

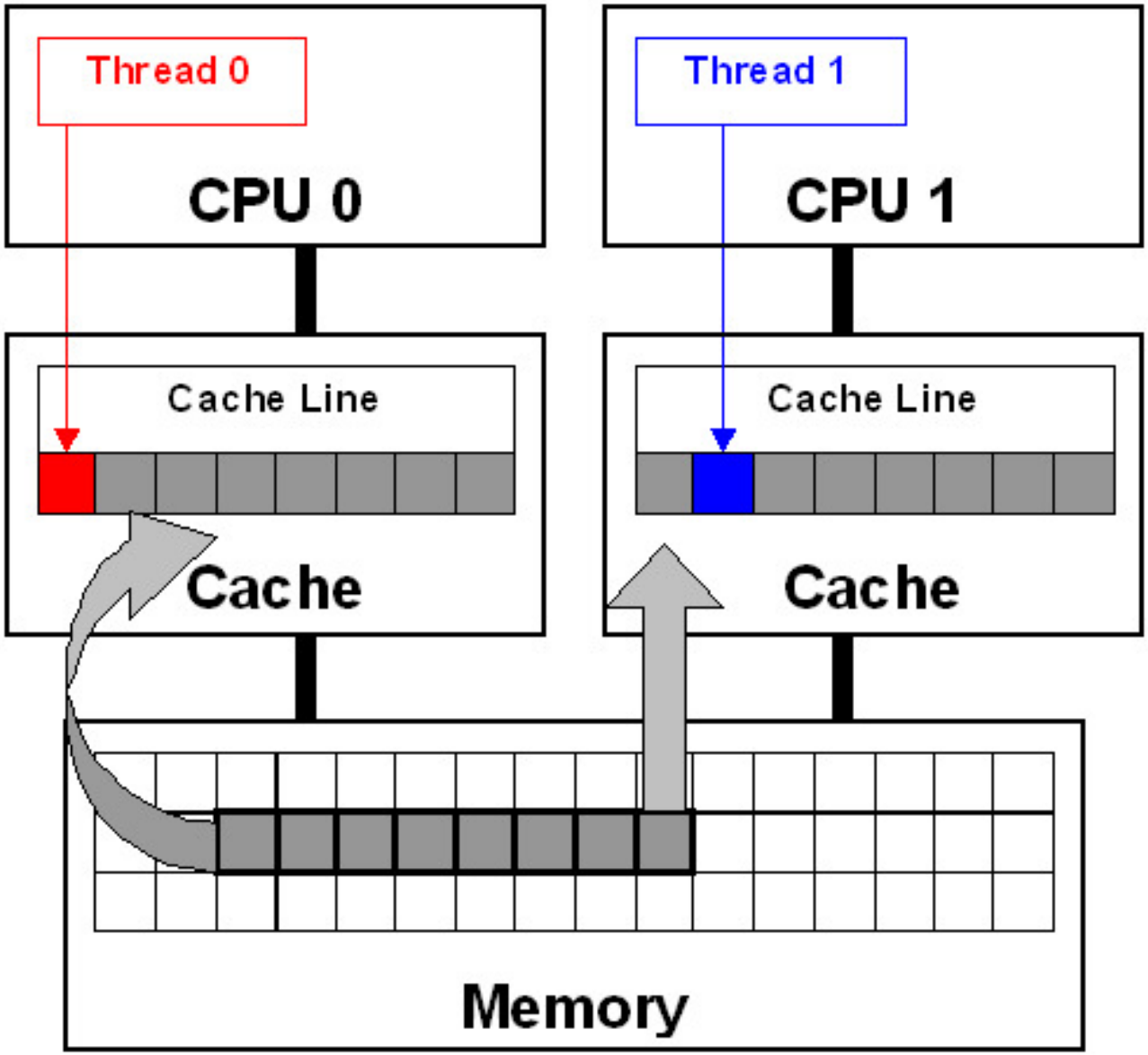
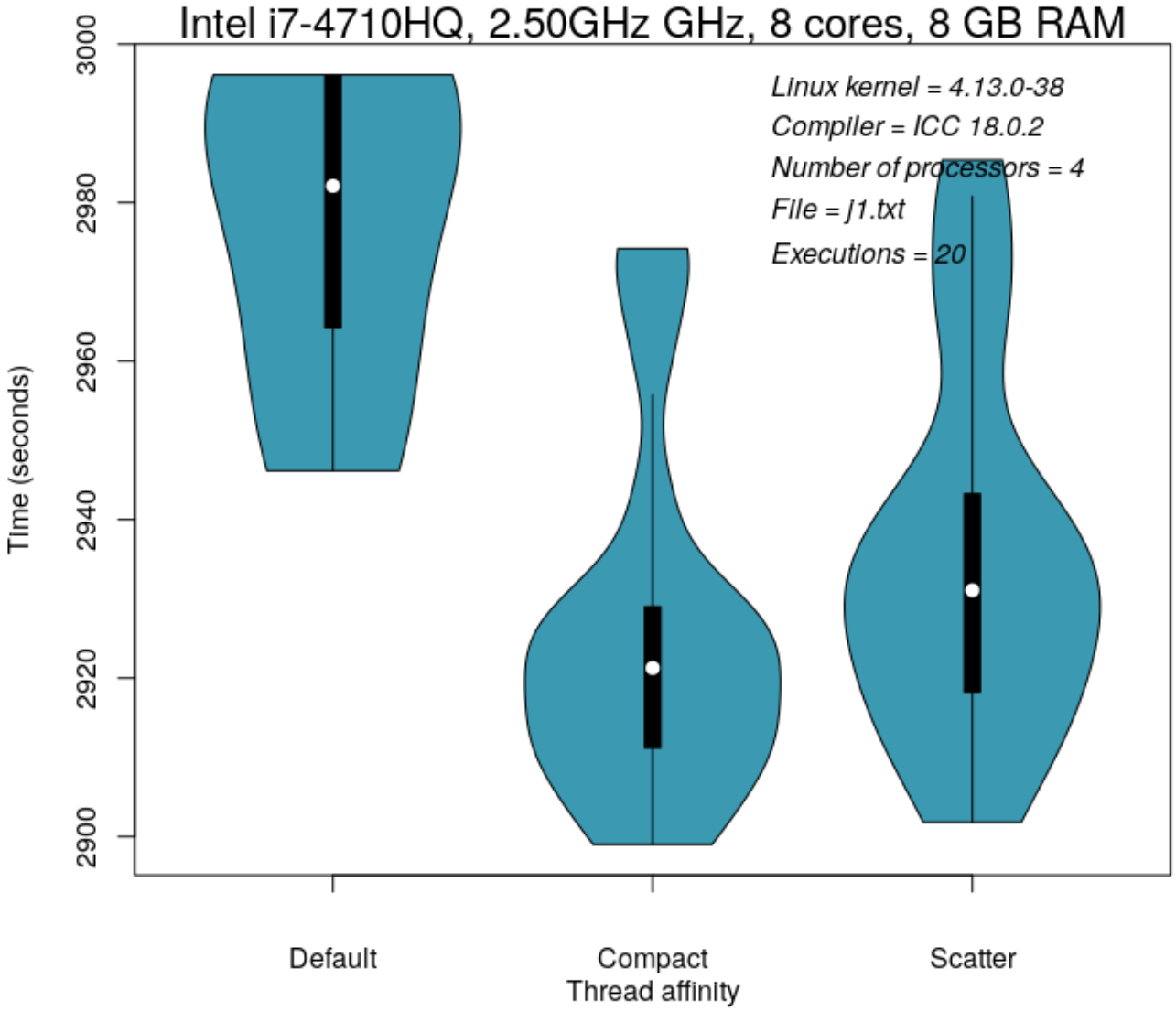


Figure: False Sharing.

Source : Intel's Documentation.

Thread Affinity

Figure: Analyzing the impact of thread placement with the Intel compiler, **-O3** with 8 CPUs.



Intel Advisor

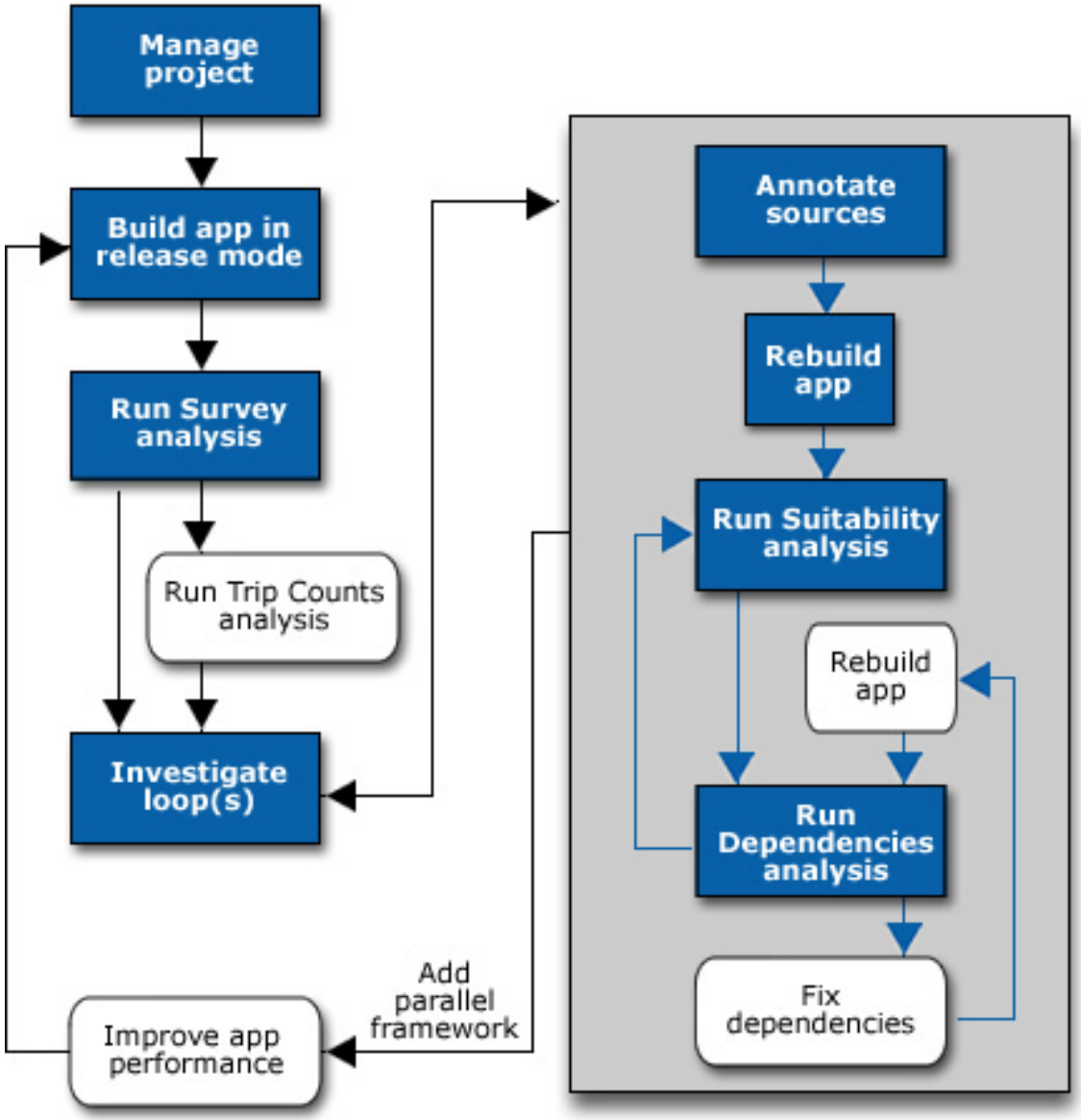


Figure: Iterative method to add parallelism (proposed by Intel).

Source : Intel's Documentation.

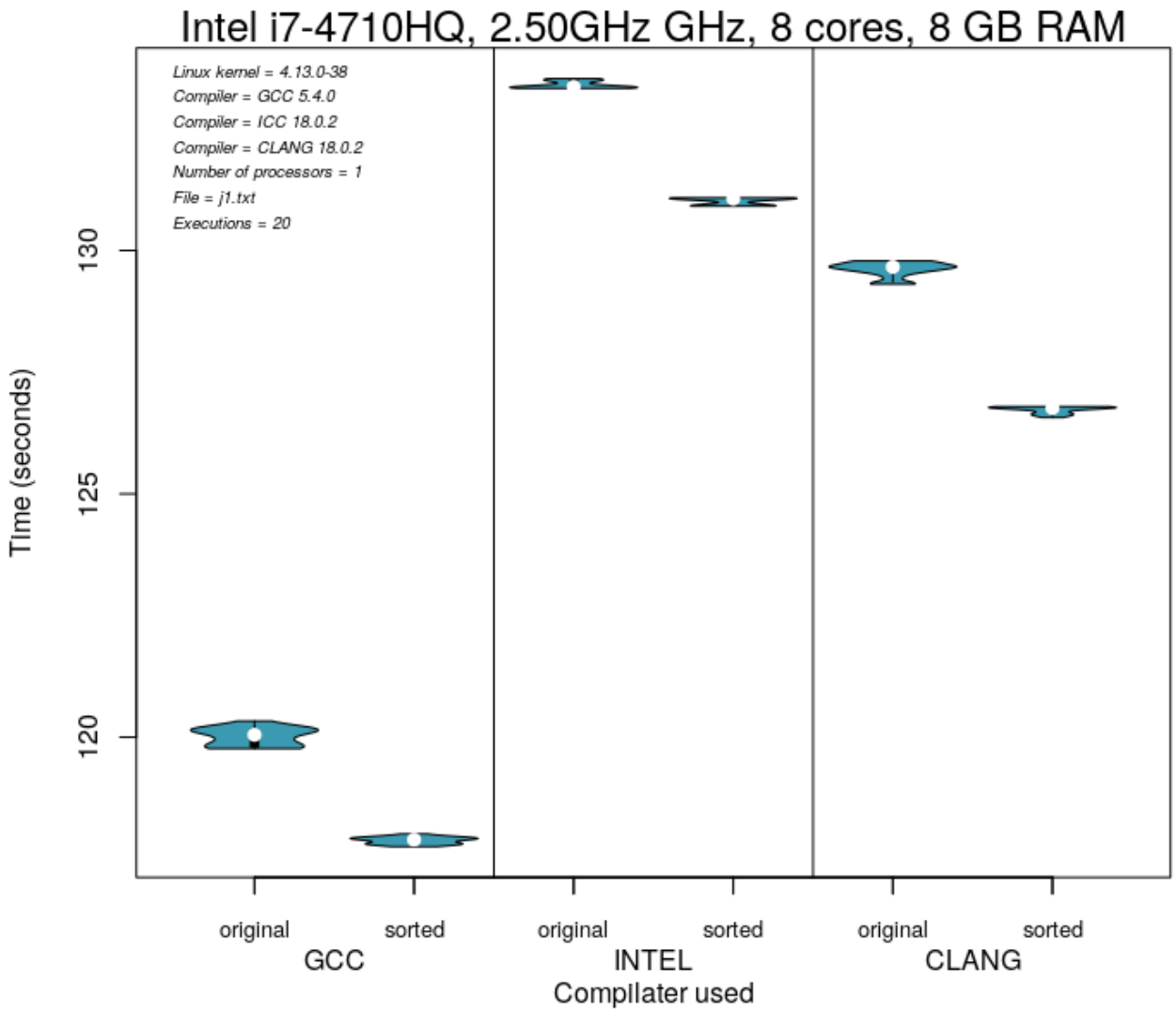
Difficulties of code optimization

- Very sequential algorithm.
- Low number of iterations ($n = 6$).
- Lot of usage of pointers that could blind the compiler's optimisation passes.

Simple algorithmic modification

- Sort the "moves" to evaluate.
- Transposition table.
- Modify algorithm to add parallelism.

Sort the "moves"



Transposition table

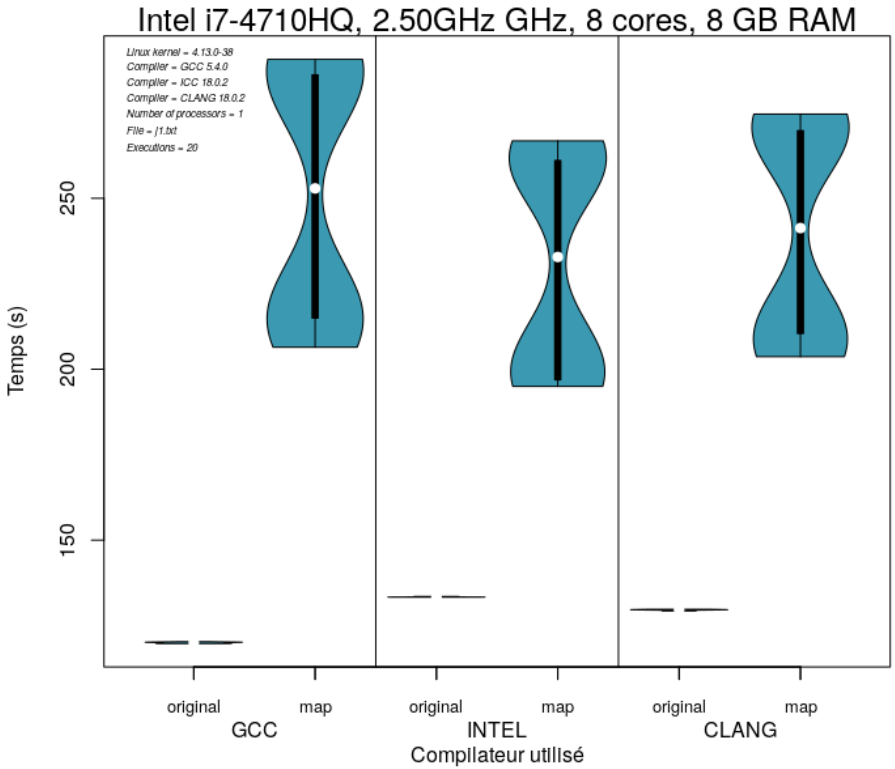


Figure: Binary search tree.

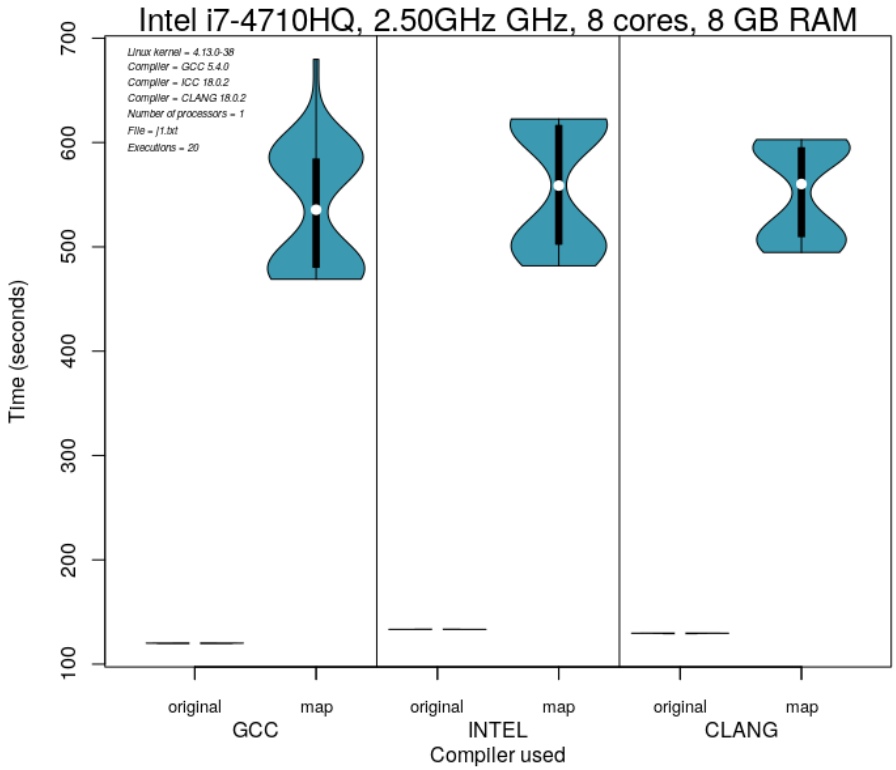
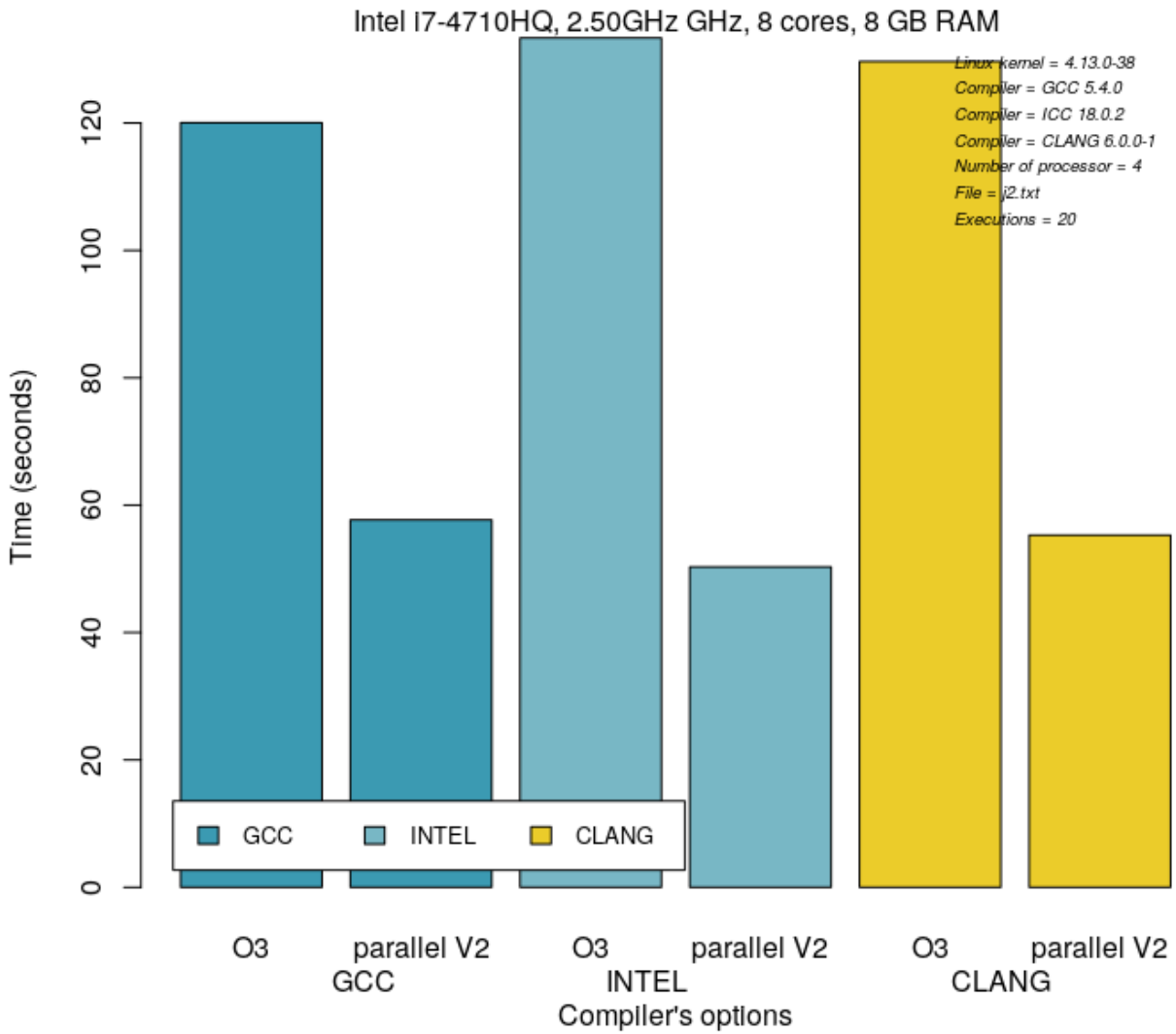


Figure: Hash table.

New parallel version



Conclusion

- GCC beats ICC for a sequential code on Intel's architecture.
- Parallelization with OpenMP can severely degrade performance.
- False Sharing phenomena.
- Algorithmic changes needed to improve performance.

Perspectives

- Thread affinity on the new version.