Automatic Grouping on Performance Investigation

Francisco de Melo

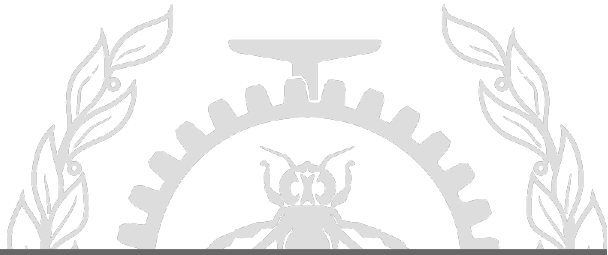
May 5th, 2017

École Polytechnique de Montréal

Laboratoire **DORSAL**

1 Outline

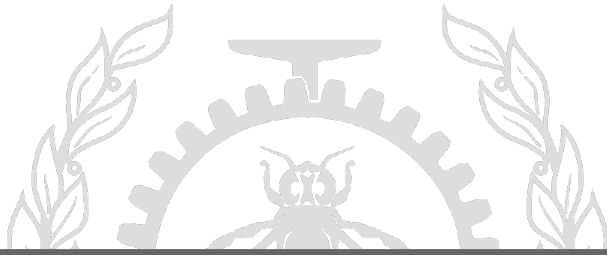
- Problem description
- ECCT Model
- Methodology
- Analysis techniques
- Utilization
- Use Case:
    - Description
    - Method
    - Results
- ECCTView
- Status



2 Problem Description

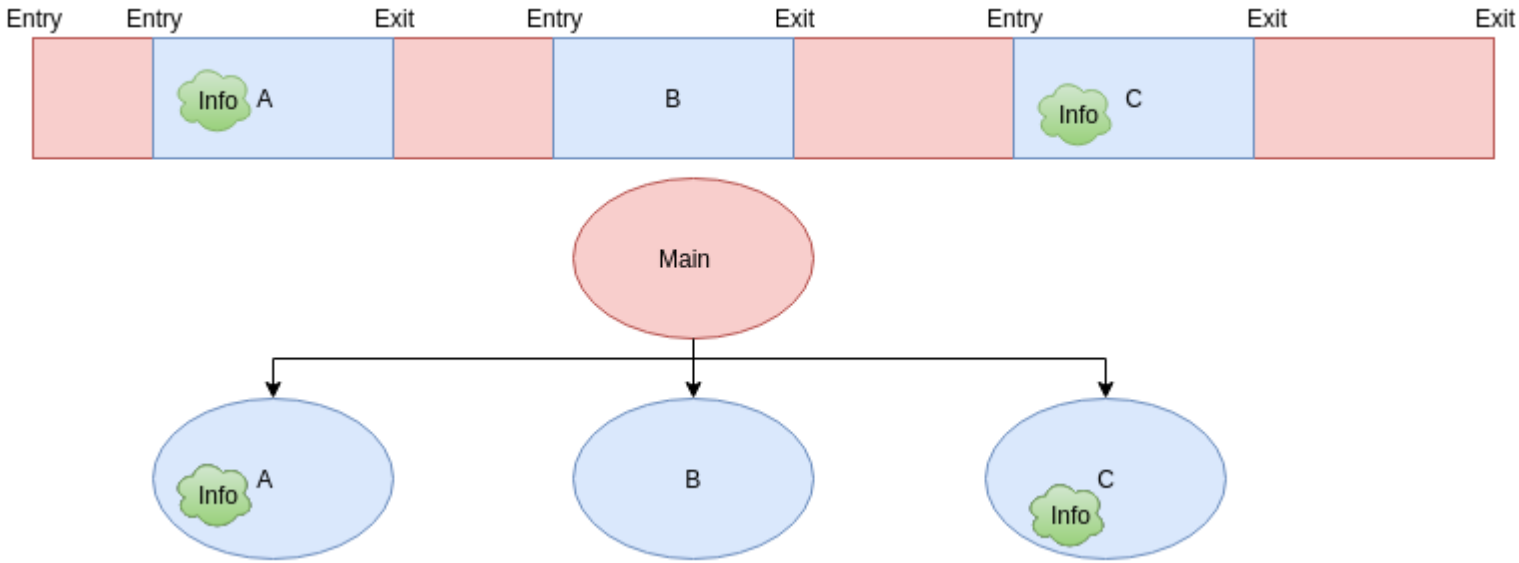
Compare **several executions** of the same software

The **current tools are limited** to compare several executions of the same software or require human analysis to **find root cause problems**.



3 ECCT Model

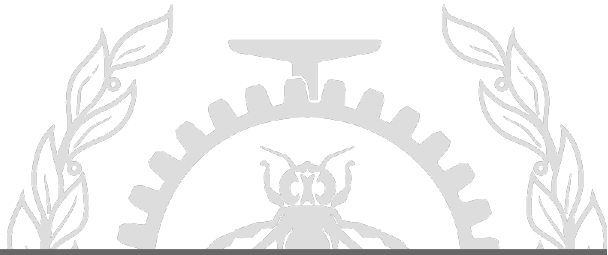
- Construction of tree using lttng and other techniques
- UST level



4 Methodology

Overall view:

- Instrument the code
- Create the tree
- Apply the classification methods

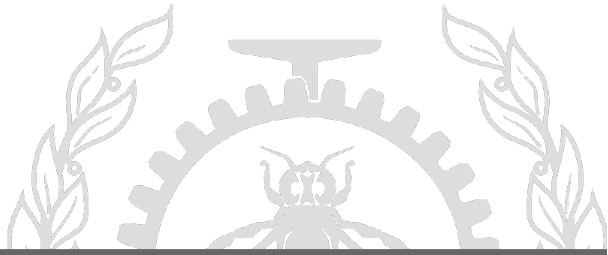


Techniques

Support Vector Machine

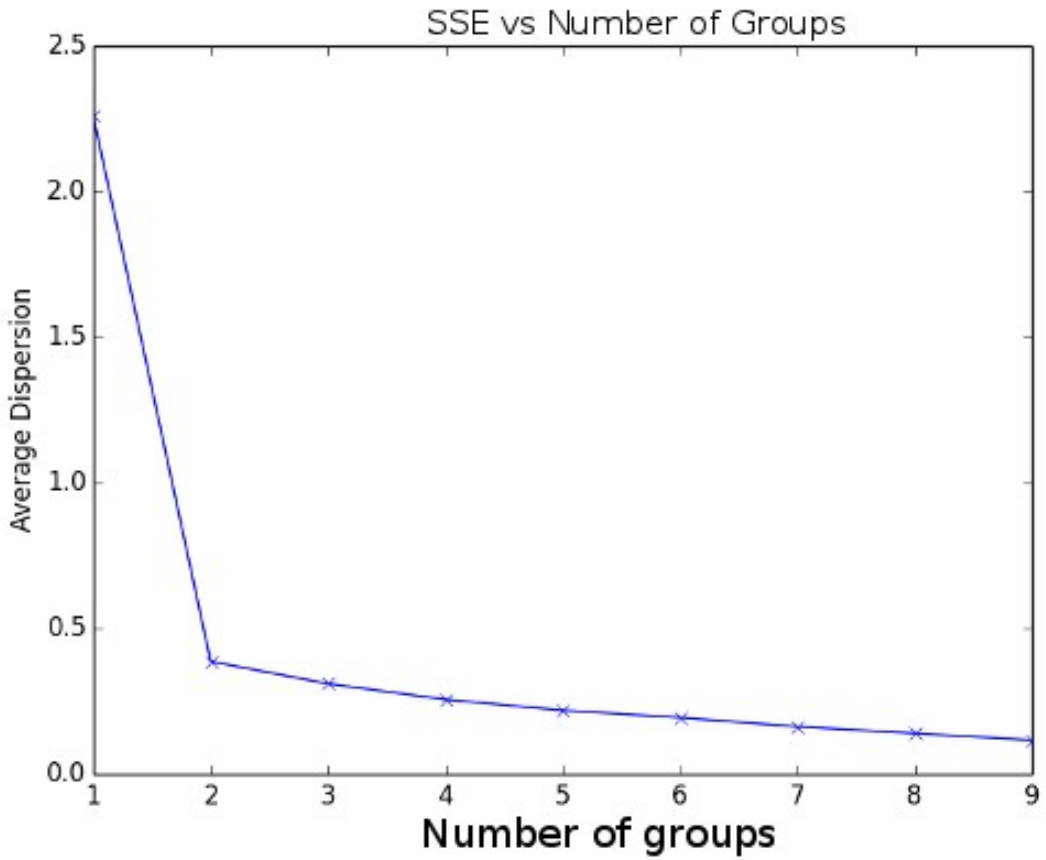
k-means algorithm

> **Auto grouping** mechanism



6 Auto grouping

Auto grouping



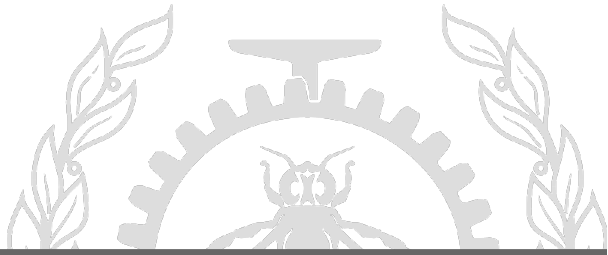
7 Utilization

This **automatic technique** can be applied for the following **situations**

Comparing C/C++ library performances

Performance Regression investigation

User Space Trace: required instrumentation



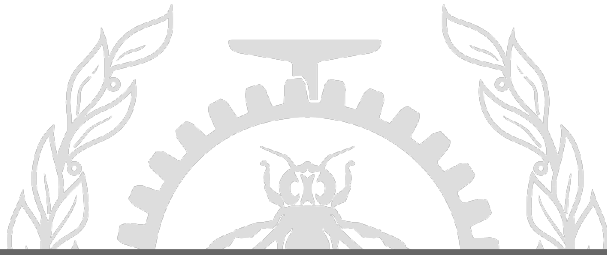
8 Real use case

OpenCV

In a regression case in OpenCV, a later version of the **HoughLines** decreased the performance in the new version.

Function: **HoughLines**

Versions: 3.1 vs 3.0



8 Real use case

HoughLines

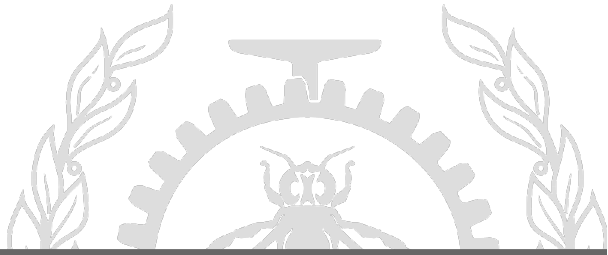
9 Approach

Run several times the process on **both** versions

Record the information **for each run**

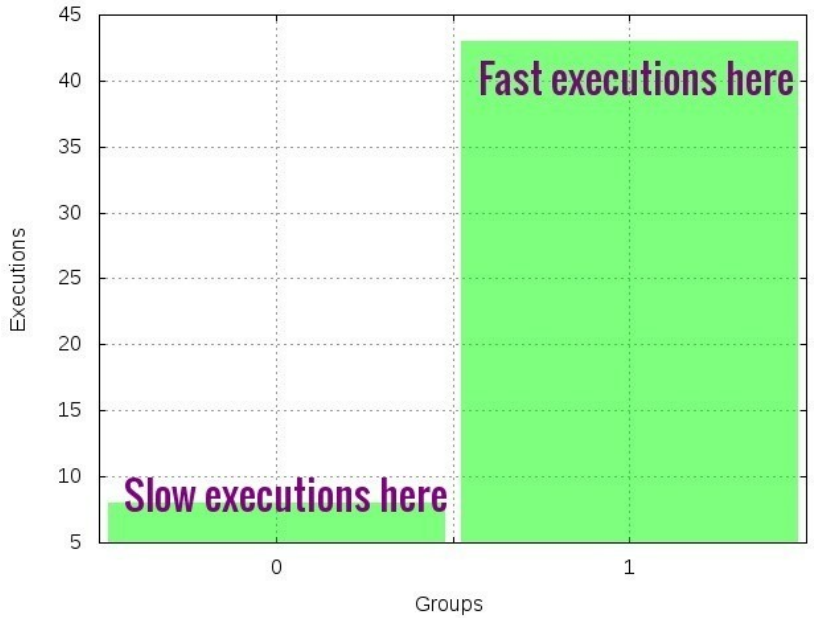
Apply the grouping techniques

Group **Comparison**



10 Results

Results: Classification in 2 groups

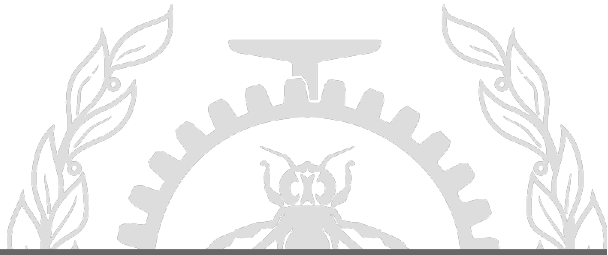


FAST Group	Slow Group
6500 Cache misses	2400 Cache misses

10 Difference

Version 3.0 to 3.1

250 commits > 4 lines of difference



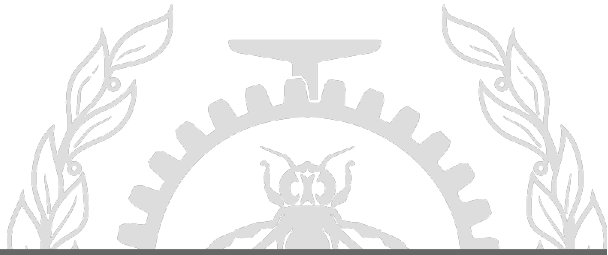
10 Conclusion

Result

Cache misses different found as performance cause by comparing two the **means of each metric**

Lines of code reduction to 4 lines

Unit test to find cache misses



11 Discussion

Automate Grouping

Take the best number of groups

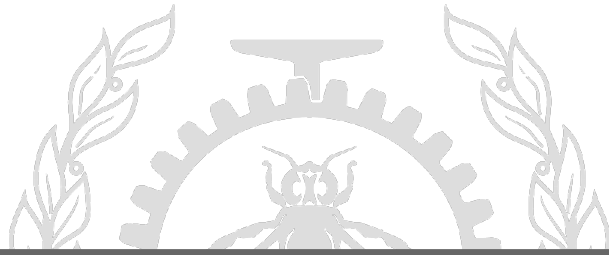
Apriori Algorithm for more complex cases

Cross over of the groups to find root causes

	A	B	C
A	X	75%	100%
B	75%	X	65%
C	100%	65%	X

Comparison of groups

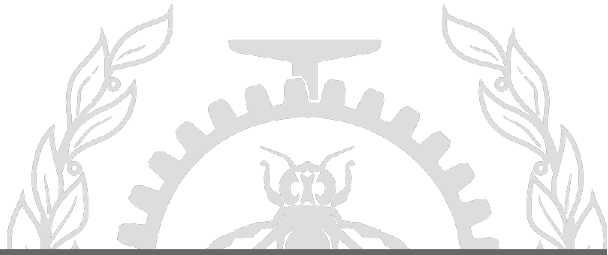
Compare them using a specific algorithm



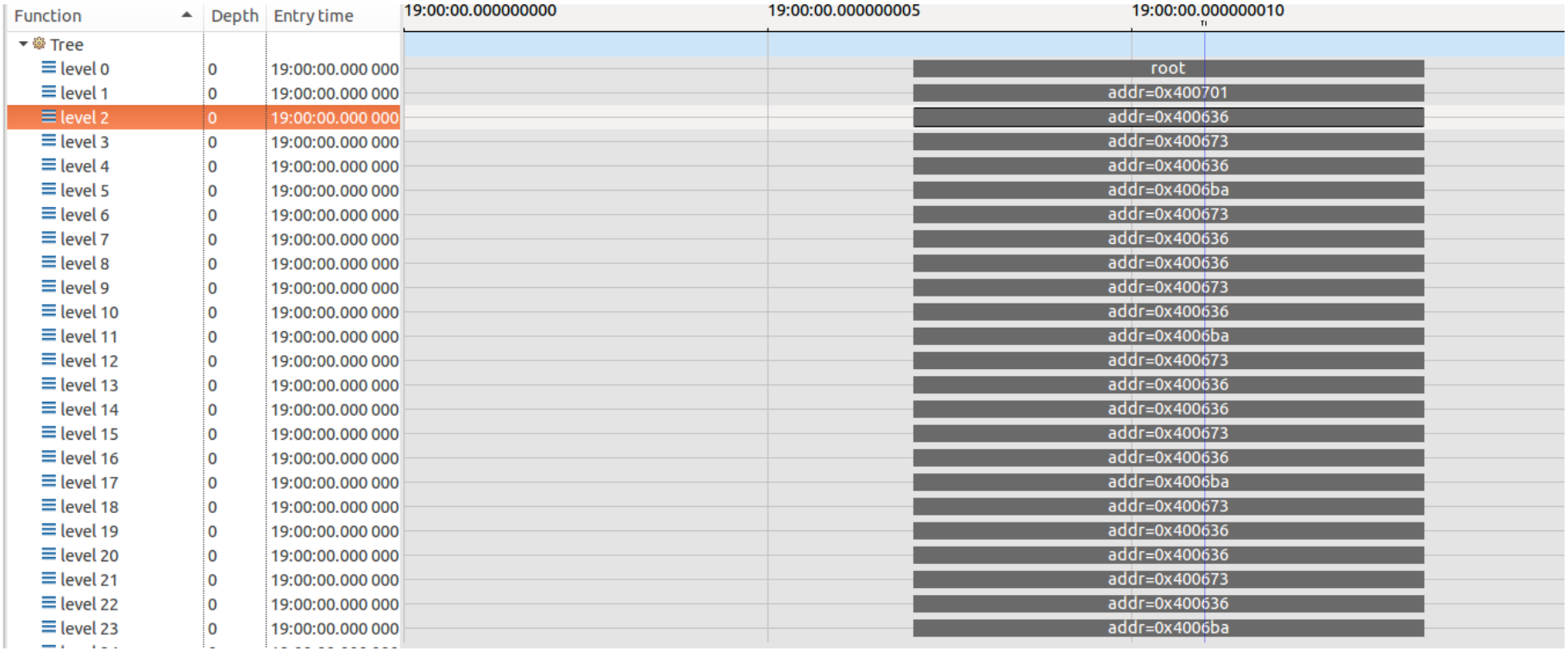
ECCT View

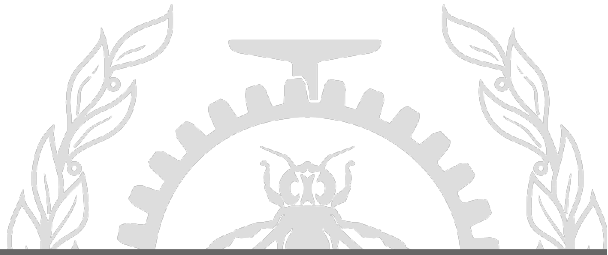
(i) Display the tree

(ii) Display groups mechanisms



12 View


Function
Depth
Entry time
19:00:00.000000000
19:00:00.000000005
19:00:00.000000010
Tree
level 0 0 19:00:00.000 000 root
level 1 0 19:00:00.000 000 addr=0x400701
level 2 0 19:00:00.000 000 addr=0x400636
level 3 0 19:00:00.000 000 addr=0x400673
level 4 0 19:00:00.000 000 addr=0x400636
level 5 0 19:00:00.000 000 addr=0x4006ba
level 6 0 19:00:00.000 000 addr=0x400673
level 7 0 19:00:00.000 000 addr=0x400636
level 8 0 19:00:00.000 000 addr=0x400636
level 9 0 19:00:00.000 000 addr=0x400673
level 10 0 19:00:00.000 000 addr=0x400636
level 11 0 19:00:00.000 000 addr=0x4006ba
level 12 0 19:00:00.000 000 addr=0x400673
level 13 0 19:00:00.000 000 addr=0x400636
level 14 0 19:00:00.000 000 addr=0x400636
level 15 0 19:00:00.000 000 addr=0x400673
level 16 0 19:00:00.000 000 addr=0x400636
level 17 0 19:00:00.000 000 addr=0x4006ba
level 18 0 19:00:00.000 000 addr=0x400673
level 19 0 19:00:00.000 000 addr=0x400636
level 20 0 19:00:00.000 000 addr=0x400636
level 21 0 19:00:00.000 000 addr=0x400673
level 22 0 19:00:00.000 000 addr=0x400636
level 23 0 19:00:00.000 000 addr=0x4006ba



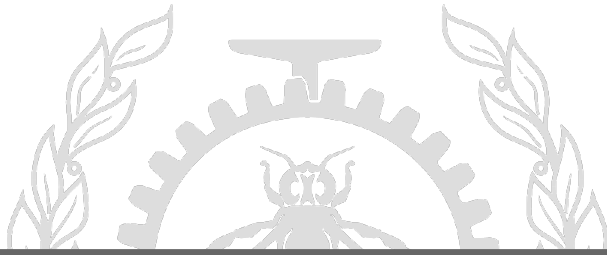
13 Status

Implemented the Tree [done]

Application of Grouping methods [done]

Apriori Method [done]

Displaying the groups and associations [doing]



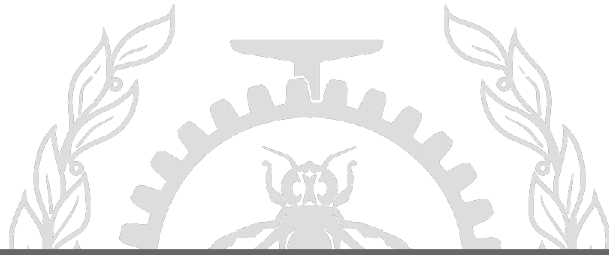
References

Doray, F, and M. R. Dagenais, "**Diagnosing Performance Variations by Comparing Multi-Level Execution Traces**", IEEE Transactions on Parallel and Distributed Systems, vol. pp,issue: 99 no. 1, 2016.

Andrea Adamoli and Matthias Hauswirth. **Trevis: A context tree visualization & analysis framework and its use for classifying performance failure reports**. In SoftVis '10: Proceedings of the ACM Symposium on Software Visualization, 2010.

J. M. Spivey. **Fast, Accurate Call Graph Profiling**. Softw. Pract. Exper., 34(3):249-264, 2004.

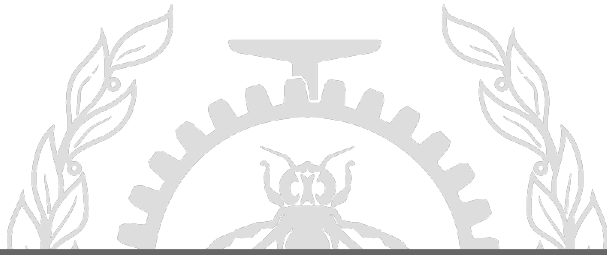
W. N. Sumner, Y. Zheng, D. Weeratunge, and X. Zhang. Precise Calling Context Encoding. In ACM International Conference on Software Engineering, 2010.



Questions

isnaldo-francisco.de-melo-jr@polymtl.ca

Any other info?



Obrigado