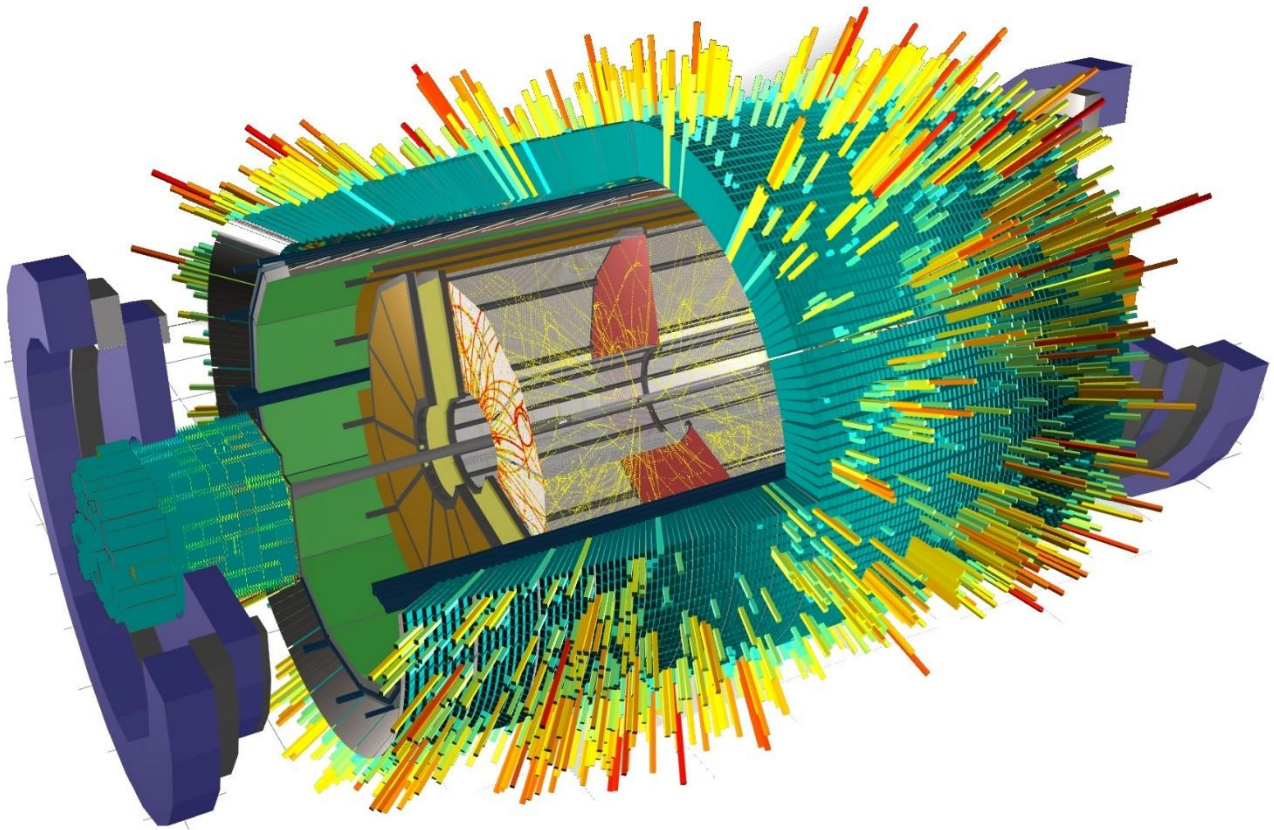
Performance Analysis and Optimization of MPDRoot

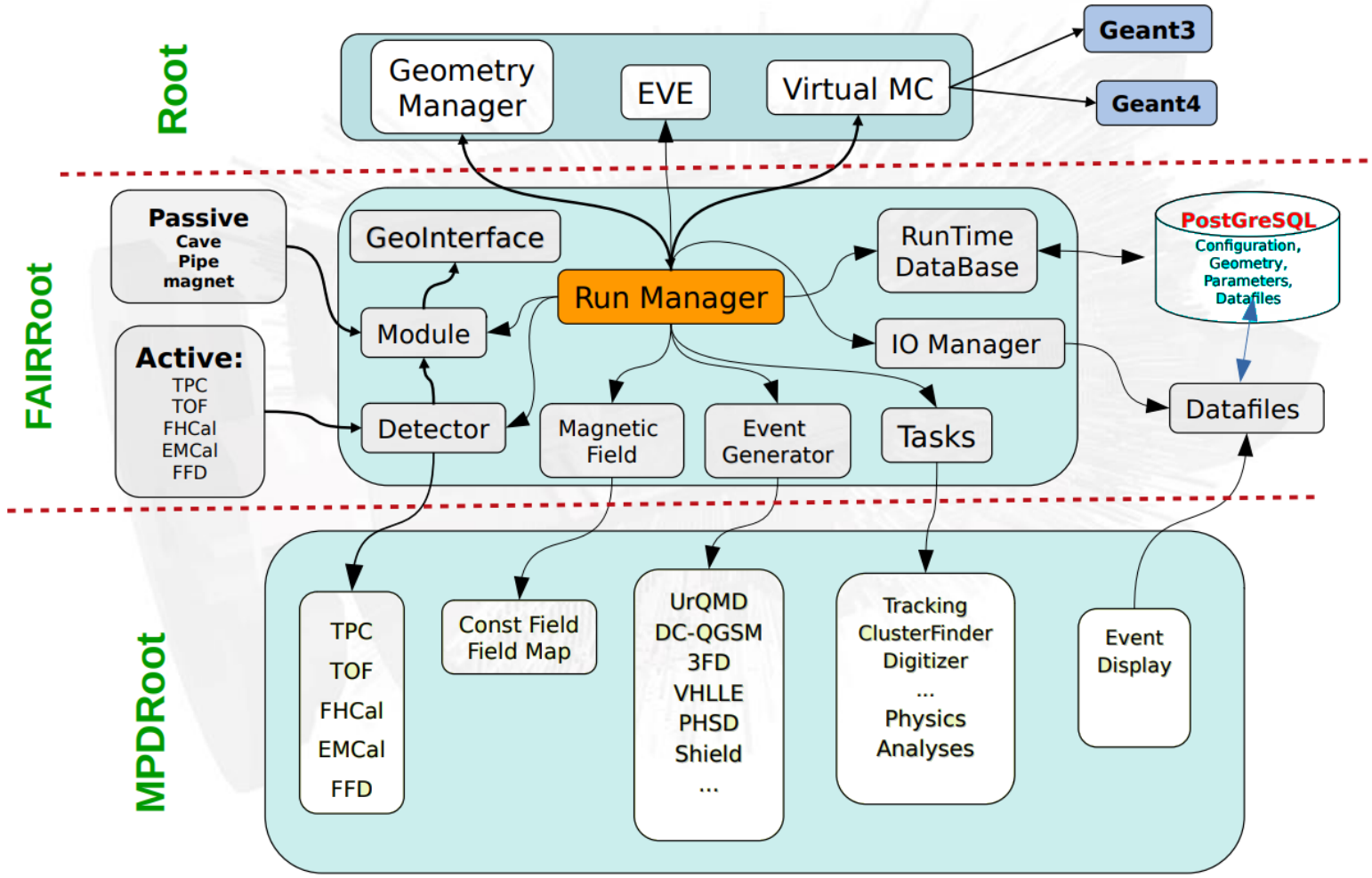
Buša J., Hnatič S., Rogachevsky O.
LIT JINR, LHEP JINR



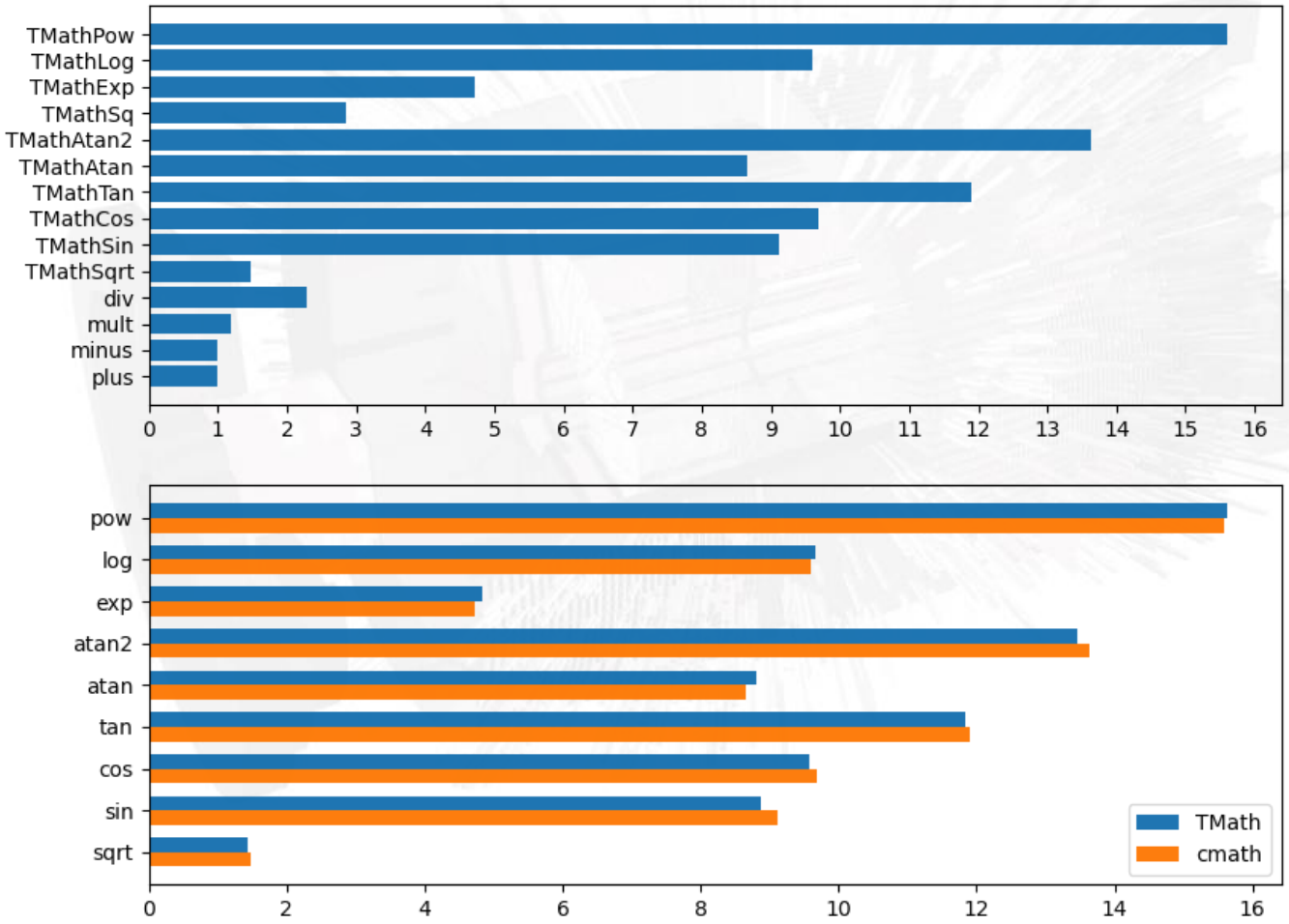
OUTLINE

- MPDRoot
- Benchmarking TMath (cmath) in MPDRoot
- Profiling MPDRoot – Instructions vs Timings
- TMath Optimization
- Reducing Calls
- Premature Optimization
- When to optimize?
- Future Perspectives

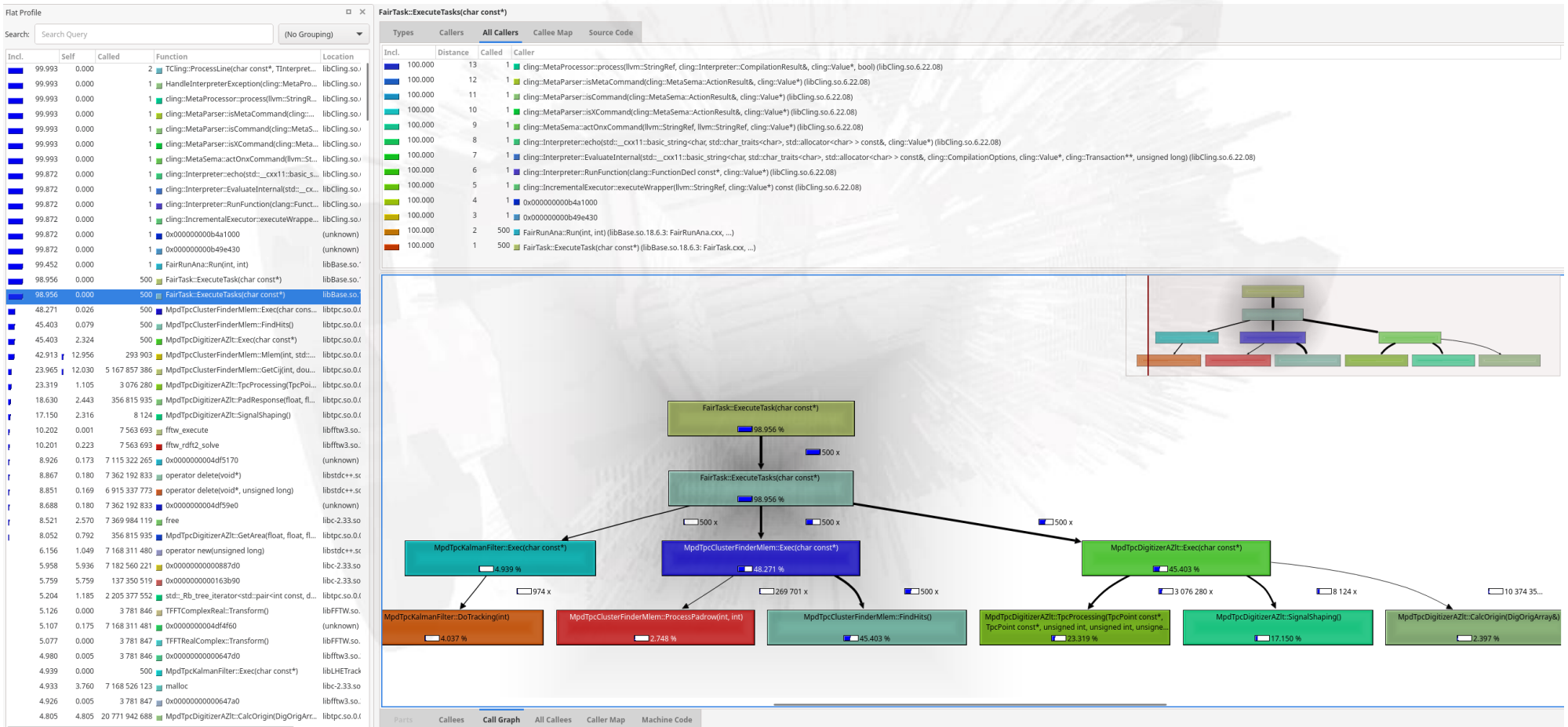
MPDRoot STRUCTURE



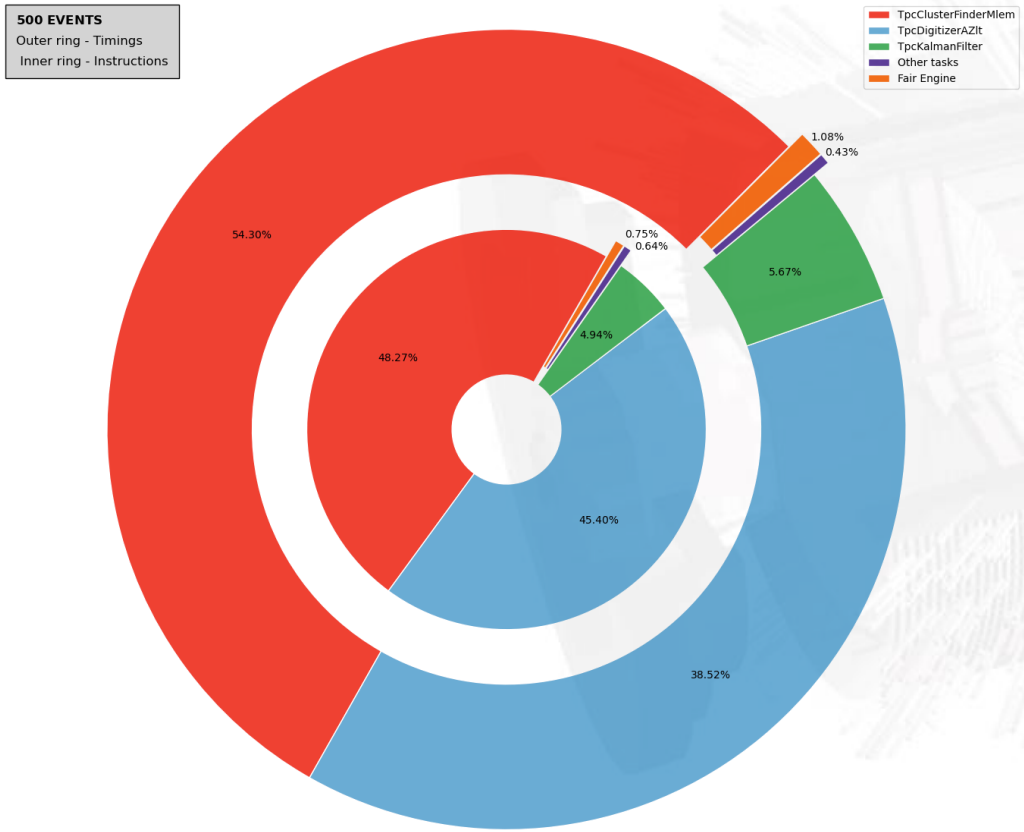
TMATH BENCHMARKS

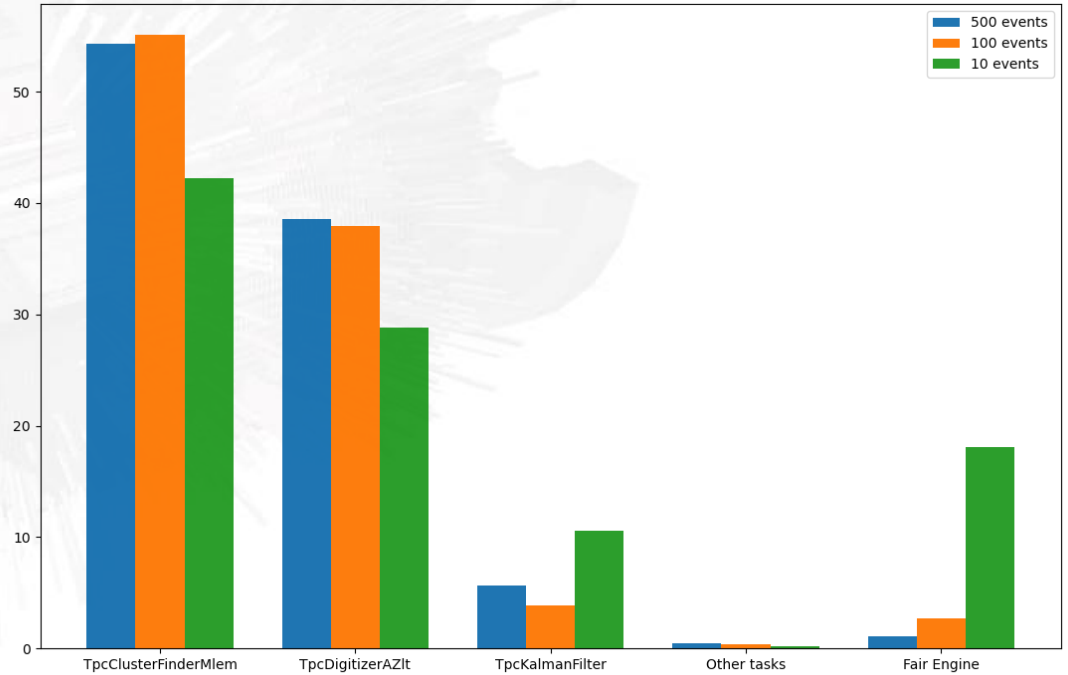


INSTRUCTION PROFILING

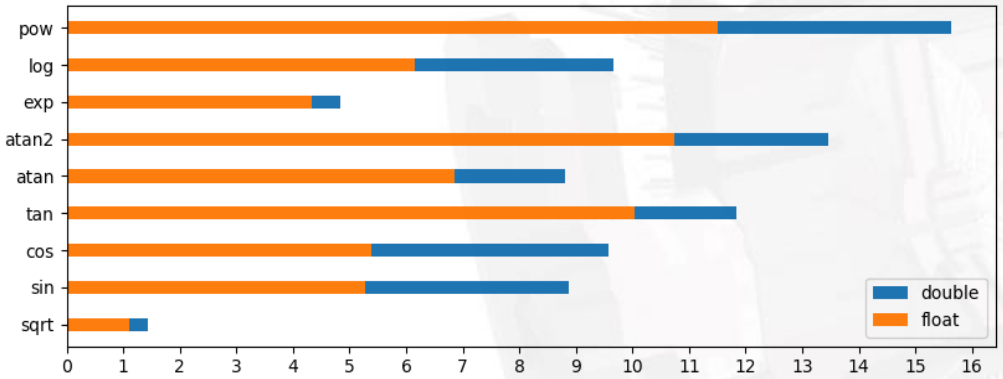


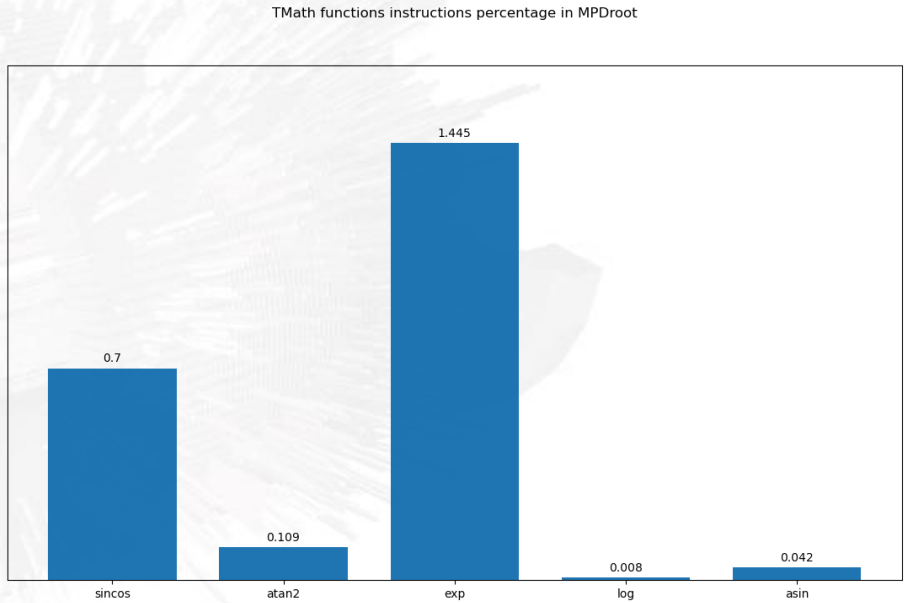
TIME PERFORMANCE





OPTIMIZATION TMATH





REDUCING CALLS

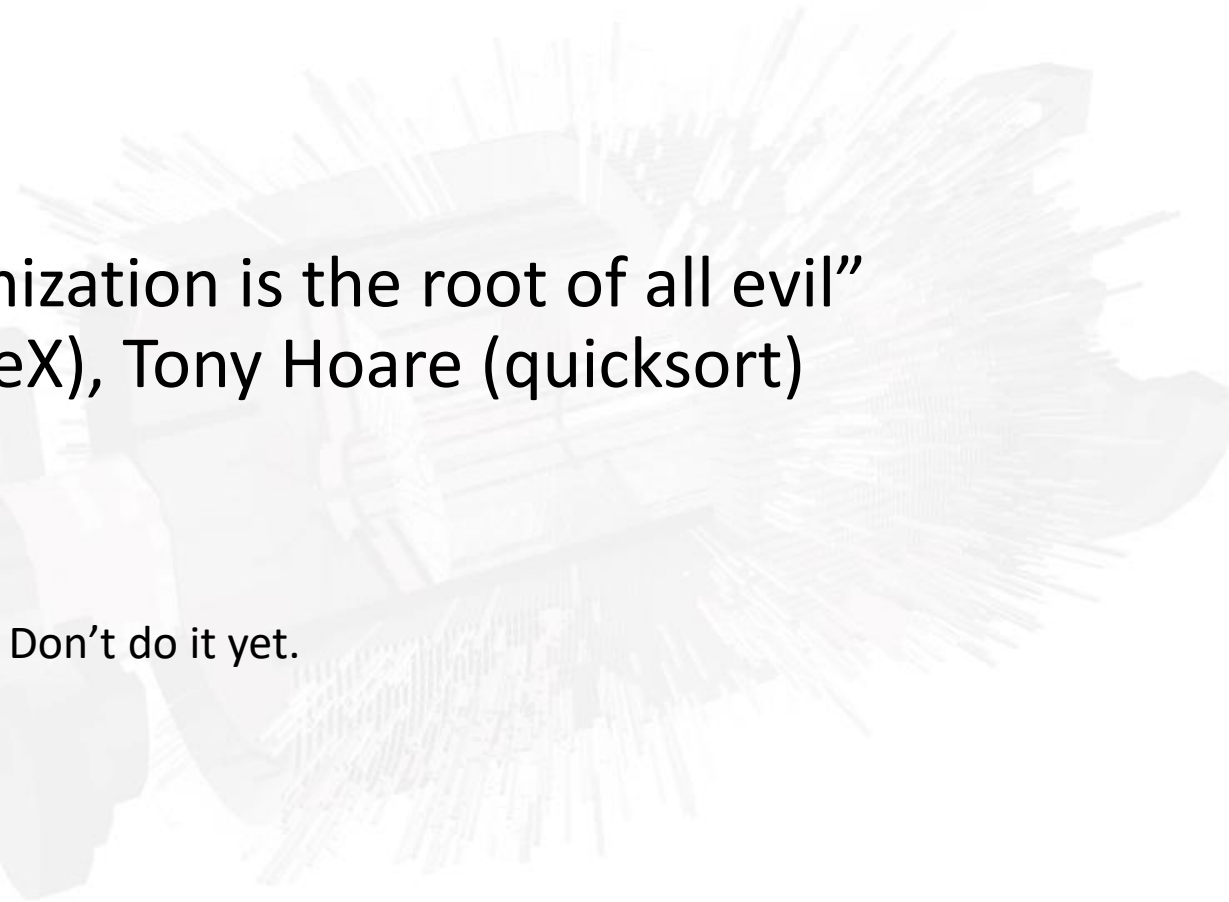
- Algorithm logic improvement
- Inlining (inline, flatten)

"Q: Do inline functions improve performance?
A: Yes and no. Sometimes. Maybe."
isocpp.org FAQ

Example from MPDRoot codebase

	% of Instructions out of total	Instructions per call	% of calls inlined	Task speedup	Total speedup
CalcOrigin (Digitizer Task)	4.8	18	100	4.2%	1.9%
GetCij (ClusterFinder Task)	12	380	90	-1.2%	-6.3%

PREMATURE OPTIMIZATION



"Premature optimization is the root of all evil"
Donald Knuth (TeX), Tony Hoare (quicksort)

Rules of optimization:
Rule 1: Don't do it.
Rule 2 (for experts only): Don't do it yet.

WHEN TO OPTIMIZE

Single responsibility principle

Open/Closed principle

Software elements (modules, classes, functions etc) should be open for extension, but closed for modification

Liskov Substitution principle

Interface Segregation principle

Dependency Inversion principle

NEAR FUTURE PERSPECTIVES

"The art of programming (software development) is the art of organizing complexity, of mastering multitude and avoiding its bastard chaos as effectively as possible."

E. Dijkstra

GETTING THE SD PROCESS UNDER CONTROL

- Code Ownership within a GIT (critical, must be developed)
- Testing environment - Improve / Automatize QA System within MPDRoot

Thank You !

Q & A