Spreadsheet Calculation Threading Improvements

Caolán McNamara
Principal Software Engineer
caolan.mcnamara@collabora.com

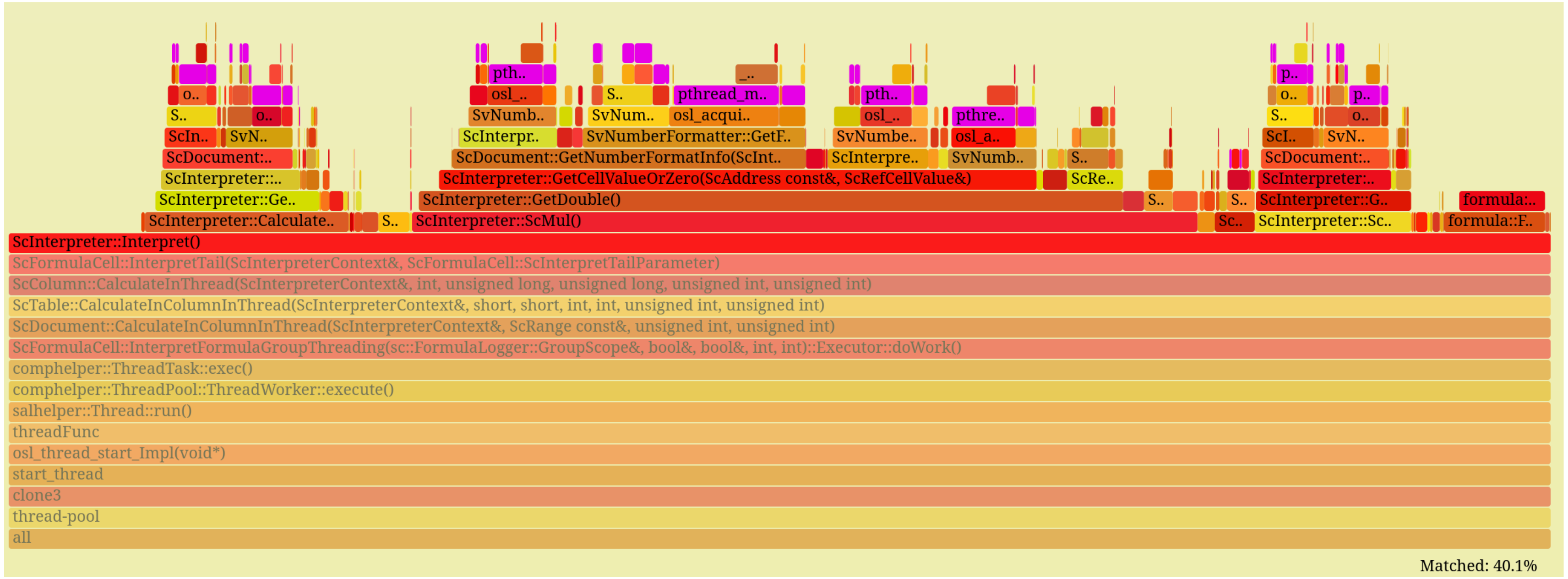
Collabora Online

LibreOffice Technology

Technical Day
COOL days

Threading Improvements

Profile to see where we spend the time. 40% in pthread_mutex_lock



Threading Improvements

Not getting full advantage of threading

- sufficient lock contention between threads that we end up bottlenecked on mutexes
- SvNumberFormatter::GetFormatForLanguageIfBuiltIn features highly

Technical Day

Threading Improvements

Some pieces that matter

- ScInterpreters are created and destroyed frequently
- Longer lived ScInterpreterContext, one for each thread, which are reused by handing one out to the short-lived ScInterpreter for its state
- Ideally ScInterpreters can execute simultaneously without anything locked
- But the document has a single NumberFormatter which has to assume it might be written to so all entry points take a mutex

Collabora Online

Threading Improvements

SvNumberFormatter

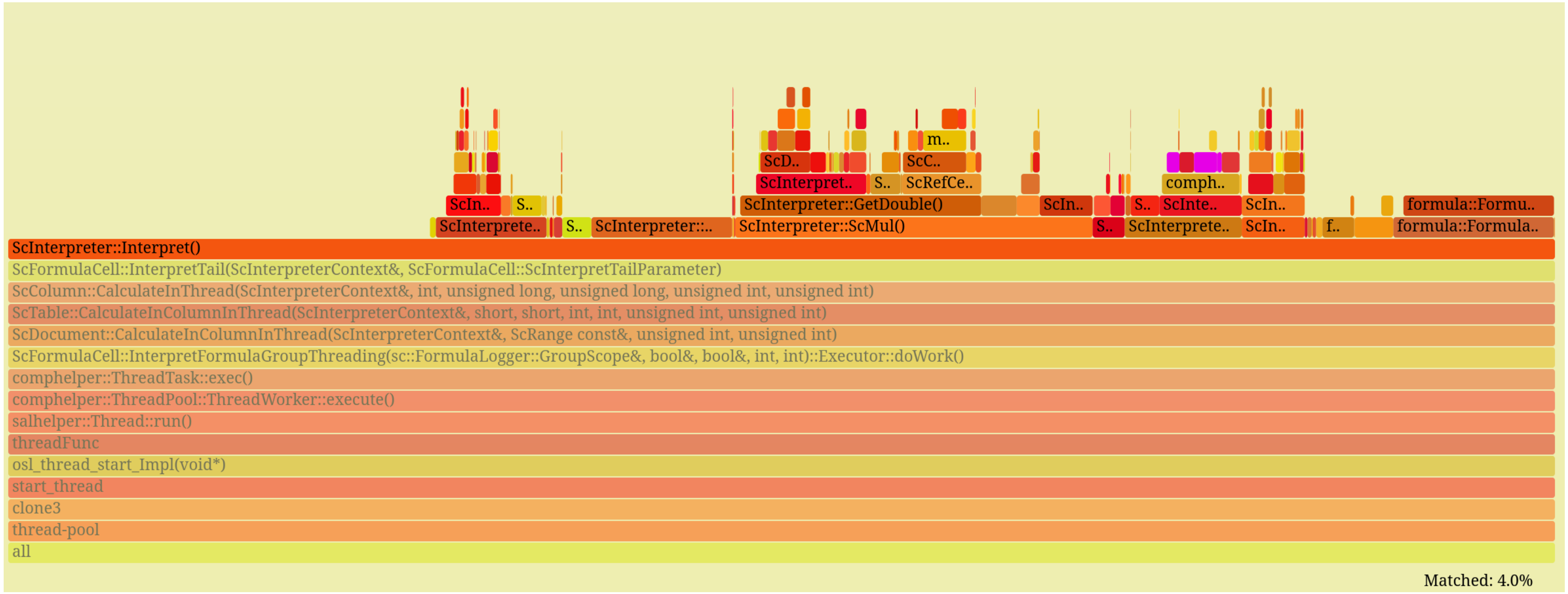
- SvNumberFormatter::GetFormatForLanguageIfBuiltIn doesn’t really do a whole lot, takes two integers and returns another.
- Should return the same thing every time, at least for the duration of interpreting the formulas
- Quick and Dirty check shows there are just four different arg combinations
- There is also a suspiciously similar-looking case of SvNumberFormatter::GetType which has a per-InterpreterContextCache of the last query
- So do similar cache of last 4 GetFormatForLanguageIfBuiltIn

Technical Day

COOL days

Threading Improvements

Profile to see where we spend the time. 4% in pthread_mutex_lock



Collabora Online

Threading Improvements

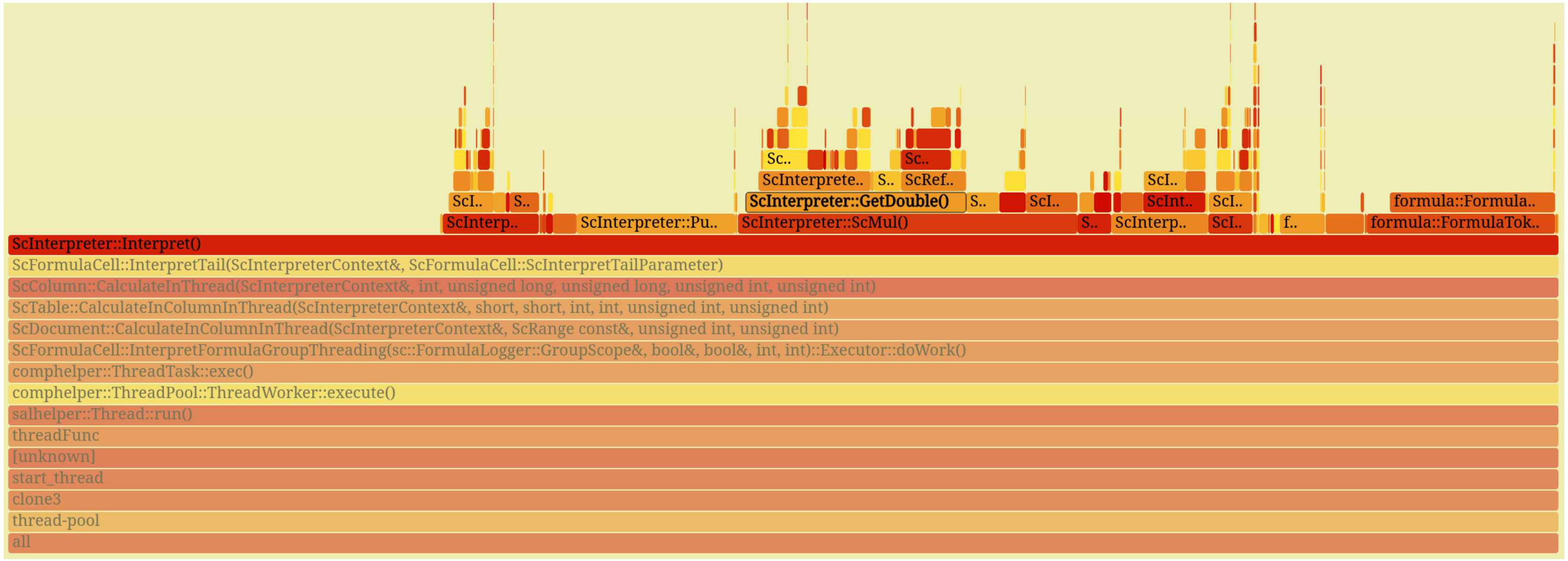
ScRandom

- Still got some locks in ScRandom
    - comphelper::uniform_real_distribution
    - single std::mt19937 generator
- Put a std::mt19937 (seeded by the global one) per InterpreterContext

Technical Day

Threading Improvements

Post ScRandom mutex removal

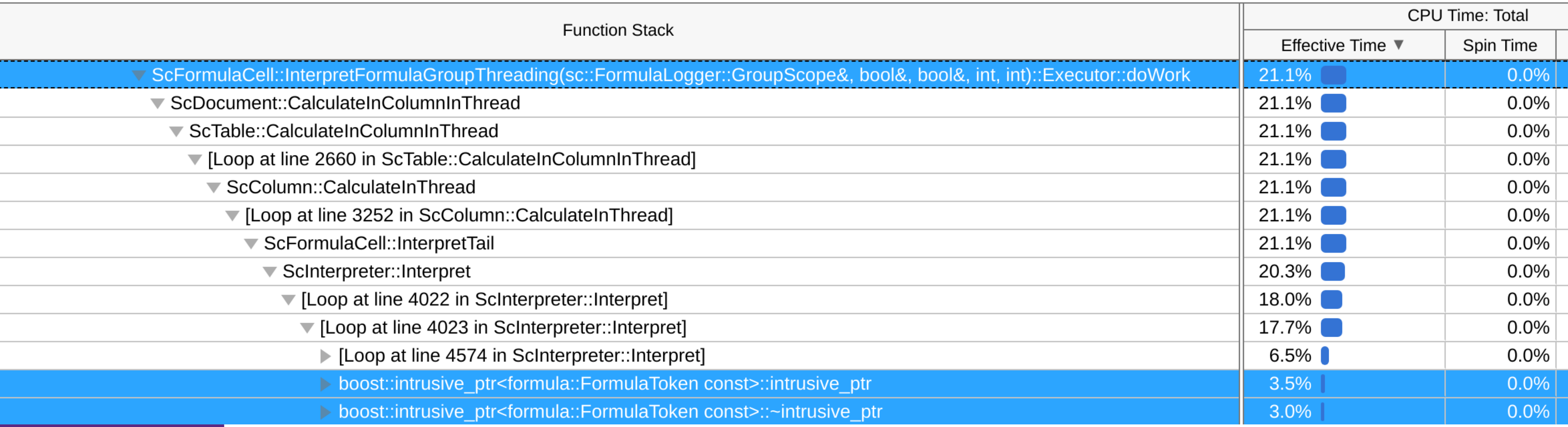


Collabora Online

Threading Improvements

Look more fine grained now

- What is this intrusive_ptr ctor/dtor pair?

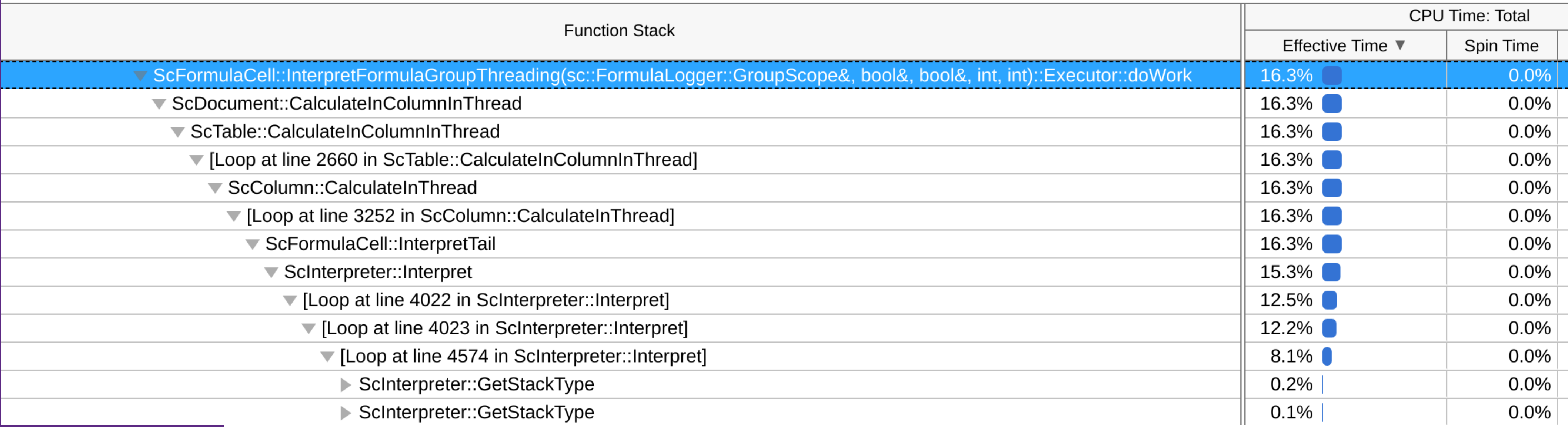


Function Stack	CPU Time: Total	
	Effective Time ▼	Spin Time
▼ ScFormulaCell::InterpretFormulaGroupThreading(sc::FormulaLogger::GroupScope&, bool&, bool&, int, int)::Executor::doWork	21.1%	0.0%
▼ ScDocument::CalculateInColumnInThread	21.1%	0.0%
▼ ScTable::CalculateInColumnInThread	21.1%	0.0%
▼ [Loop at line 2660 in ScTable::CalculateInColumnInThread]	21.1%	0.0%
▼ ScColumn::CalculateInThread	21.1%	0.0%
▼ [Loop at line 3252 in ScColumn::CalculateInThread]	21.1%	0.0%
▼ ScFormulaCell::InterpretTail	21.1%	0.0%
▼ ScInterpreter::Interpret	20.3%	0.0%
▼ [Loop at line 4022 in ScInterpreter::Interpret]	18.0%	0.0%
▼ [Loop at line 4023 in ScInterpreter::Interpret]	17.7%	0.0%
▶ [Loop at line 4574 in ScInterpreter::Interpret]	6.5%	0.0%
▶ boost::intrusive_ptr<formula::FormulaToken const>::intrusive_ptr	3.5%	0.0%
▶ boost::intrusive_ptr<formula::FormulaToken const>::~intrusive_ptr	3.0%	0.0%

Threading Improvements

Look more fine grained now

- Creating a temporary to search for an intrusive_ptr key in map
- C++ 14 has a feature designed for just this type of issue



Function Stack	CPU Time: Total	
	Effective Time ▼	Spin Time
▼ ScFormulaCell::InterpretFormulaGroupThreading(sc::FormulaLogger::GroupScope&, bool&, bool&, int, int)::Executor::doWork	16.3%	0.0%
▼ ScDocument::CalculateInColumnInThread	16.3%	0.0%
▼ ScTable::CalculateInColumnInThread	16.3%	0.0%
▼ [Loop at line 2660 in ScTable::CalculateInColumnInThread]	16.3%	0.0%
▼ ScColumn::CalculateInThread	16.3%	0.0%
▼ [Loop at line 3252 in ScColumn::CalculateInThread]	16.3%	0.0%
▼ ScFormulaCell::InterpretTail	16.3%	0.0%
▼ ScInterpreter::Interpret	15.3%	0.0%
▼ [Loop at line 4022 in ScInterpreter::Interpret]	12.5%	0.0%
▼ [Loop at line 4023 in ScInterpreter::Interpret]	12.2%	0.0%
▼ [Loop at line 4574 in ScInterpreter::Interpret]	8.1%	0.0%
▶ ScInterpreter::GetStackType	0.2%	0.0%
▶ ScInterpreter::GetStackType	0.1%	0.0%

Technical Day

Threading Improvements

ReadOnly NumberFormatter

- During calculation we shouldn’t really need to write to the NumberFormatter
- Big refactor of NumberFormatter to break it up into the different things it does
    - Can have a ReadOnly Number Formatter mode
    - Per InterpreterContext caches that can be merged back to NumberFormatter when threading area is complete
    - Per InterpreterContext “language data” scratch data that can be discarded
- Needs default currency to be determined before use

Technical Day

COOL days

Threading Improvements

Single threaded

- ScFormulaCell::InterpretFormulaGroup **15407.025 ms**

Initial Threaded Contention

- ScFormulaCell::InterpretFormulaGroup **25997.699 ms**

Final Threaded Contention

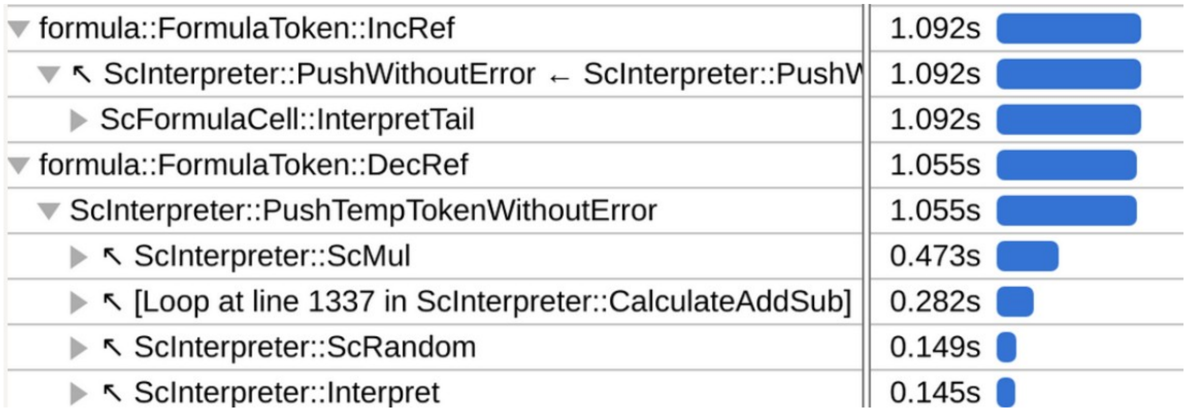
- ScFormulaCell::InterpretFormulaGroup **3215.96 ms**

Crashtesting run has gone from 3 days to 36 hours?

- 650,000 spreadsheet docs

Collabora Online

Threading Improvements



```
▼ formula::FormulaToken::IncRef                                          1.092s
  ▼ ↖ ScInterpreter::PushWithoutError ← ScInterpreter::PushW             1.092s
    ▶ ScFormulaCell::InterpretTail                                       1.092s
▼ formula::FormulaToken::DecRef                                          1.055s
  ▼ ScInterpreter::PushTempTokenWithoutError                             1.055s
    ▶ ↖ ScInterpreter::ScMul                                             0.473s
    ▶ ↖ [Loop at line 1337 in ScInterpreter::CalculateAddSub]            0.282s
    ▶ ↖ ScInterpreter::ScRandom                                          0.149s
    ▶ ↖ ScInterpreter::Interpret                                         0.145s
```

To Do

- Reference counting on FormulaTokens is expensive
- std::atomic-alike inc/dec is still a bottleneck
- Experimental approach that assume initial tokens ref counts as immutable → **2522.96 ms**

Collabora Online

LibreOffice Technology

Technical Day

COOL days

Thank you!

T S P D

@CollaboraOffice
hello@collaboraoffice.com
www.collaboraoffice.com