rosidlcpp

A Journey Through ROS 2 Build Time Optimization

Anthony Welte

https://github.com/TonyWelte/rosidlcpp

What is rosidl ?

`build/px4_msgs/.ninja_log` → ui.perfetto.dev

1 min 21 s

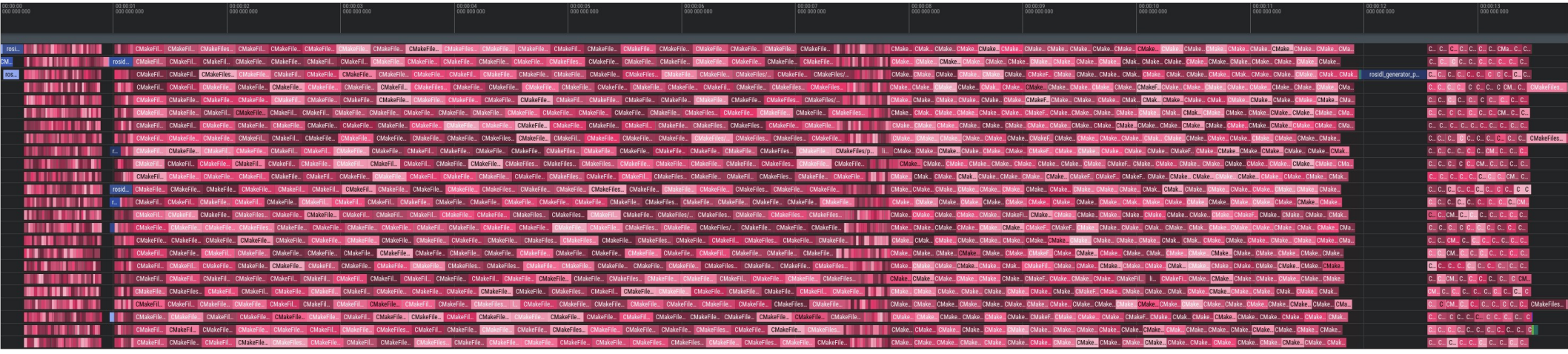
What is rosidl ?

`build/px4_msgs/.ninja_log` → ui.perfetto.dev

1 min 21 s

- 10 Python processes (rosidl generators)
- Lots of small compilation steps

C++ Implementation



13.8 s

- rosidlcpp generators are 30 to 300 times faster

gcc -ftime-report

Time variable		usr		sys		wall		GGC	
phase setup	:	0.00	(0%)	0.00	(0%)	0.00	(0%)	1882k	(2%)
phase parsing	:	**0.14**	**(88%)**	**0.11**	**(100%)**	**0.26**	**(93%)**	**68M**	**(90%)**
phase lang. deferred	:	0.01	(6%)	0.00	(0%)	0.01	(4%)	3741k	(5%)
phase opt and generate	:	0.01	(6%)	0.00	(0%)	0.01	(4%)	1746k	(2%)
|name lookup	:	0.04	(25%)	0.01	(9%)	0.04	(14%)	2962k	(4%)
|overload resolution	:	0.00	(0%)	0.00	(0%)	0.02	(7%)	3727k	(5%)
callgraph construction	:	0.01	(6%)	0.00	(0%)	0.00	(0%)	272k	(0%)
preprocessing	:	0.01	(6%)	0.04	(36%)	0.07	(25%)	2166k	(3%)
parser (global)	:	0.04	(25%)	0.02	(18%)	0.03	(11%)	25M	(34%)
parser struct body	:	0.02	(13%)	0.00	(0%)	0.02	(7%)	17M	(23%)
parser function body	:	0.01	(6%)	0.01	(9%)	0.02	(7%)	1857k	(2%)
parser inl. func. body	:	0.02	(12%)	0.00	(0%)	0.04	(14%)	2720k	(4%)
parser inl. meth. body	:	0.00	(0%)	0.02	(18%)	0.02	(7%)	8009k	(10%)
template instantiation	:	0.05	(31%)	0.02	(18%)	0.06	(21%)	14M	(19%)
constant expression evaluation	:	0.00	(0%)	0.00	(0%)	0.01	(4%)	118k	(0%)
initialize rtl	:	0.00	(0%)	0.00	(0%)	0.01	(4%)	12k	(0%)
TOTAL	:	0.16		0.11		0.28		75M	

Precompiled headers

5.8 s

- See Cmake → `target_precompile_headers`

Simplify dependencies

5.1 s

- All generators start immediately
- Contributed to rosidl#910

Thank you

```
Ninja build:
  build/px4_msgs/.ninja_log → ui.perfetto.dev

Compilation:
  gcc/clang → -ftime-report

Cmake:
  target_precompile_headers
```

https://github.com/TonyWelte/rosidlcpp