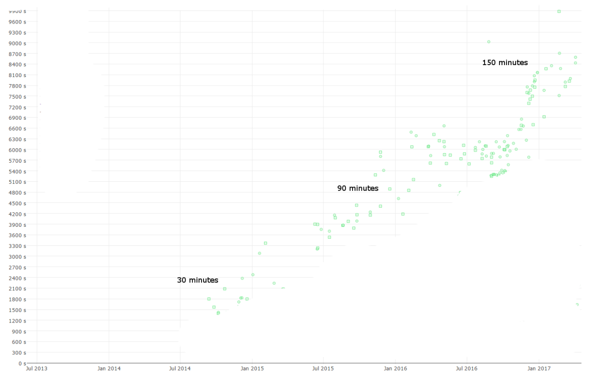
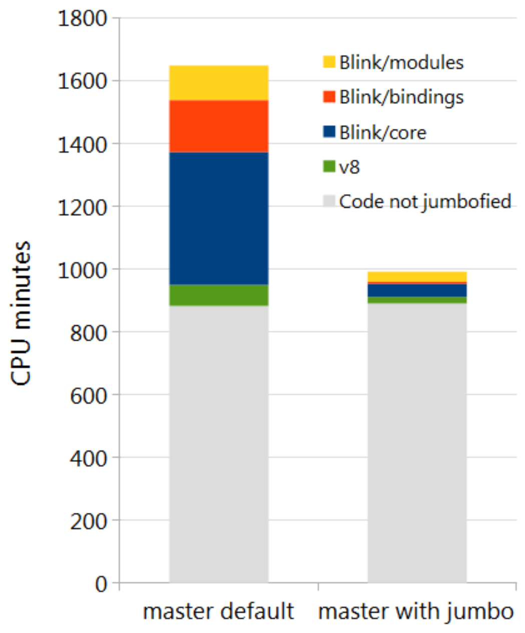
Jumbo - bringing shorter compile times to you

Background



Compiling Chromium related products have always taken a long time, but it has incrementally become worse to a point where something has to be done if the project is ever to have active external contributors. It is a stated goal of the project to reach more diverse contributors and that will not happen until the hardware and time requirements to participate are lowered.

Compiling a basic subset of Chromium today requires hundreds of CPU minutes with Blink code (specifically the code in third_party/WebKit) being about 40% of the total compilation effort.



The effect of applying the jumbo build concept to the core folder in Blink is a compile time reduction for that folder from 121 to 36 CPU minutes (-70%). That is alone a reduction by 20% of the total CPU usage during a build.

Jumbo build == unity build

The concept of a unity build is that multiple compilation units are compiled together. This works as a speedup because normally each compilation unit will repeat part of the work done by earlier compilation units. They will compile the same headers, start the same binaries, read the same files from disk, generate the same debug data. Later all this duplicated information will have to merged by linkers which is also slow.

Measurements on files in `third_party/WebKit` files show that they include about 200,000 lines of headers for each compilation unit, and on a small file that is roughly 90-95% of the compilation time. In `content`, a folder with less extreme header sizes, the includes are still around 100,000 lines.

Naming

I use the name "jumbo" build in this project to make communication easier. For a while I used the terms "unity" and "unity builds", but it turned out that the word "unity" has many overloaded related meanings which introduced confusion. "Jumbo" as a name was used for unity builds in the Presto era, and has no alternative interpretations so after switching to always calling it "jumbo", communication has been simpler.

History

In the Chromium code base this concept is already used by V8 bindings (except for Mac and Linux where it was recently disabled) and in sqlite.

At Opera we use it for our Presto based browser since 12 November 2009. People were initially skeptical but after introduction it was heartily embraced since the time savings easily made up for any added quirks (saving more than ⅔ of the compile times).

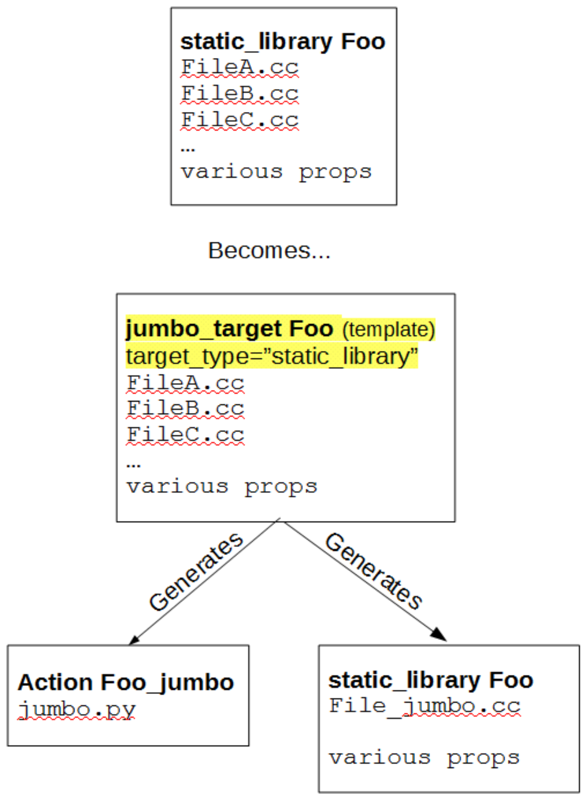
In other projects it is not common but appears every now and then. It is mostly used in large projects which have compile time problems which means that it is rarely used in public projects. Anecdotally it is common in the gaming industry.

Design

The image to the right illustrates how jumbo is implemented. The only lines that need to change are the ones marked with yellow.

The template jumbo_target will generate an action that builds the jumbo files, and a proper target that compiles the jumbo files using the same state as the original.

The action will generate files like `html_jumbo_1.cpp`, `html_jumbo_2.cpp`, `svg_jumbo_1.cpp`, ... files each containing an include statement for a proper cpp file. That is identical to how v8 bindings already compile (except in Linux and Mac where that was recently disabled).



Size of jumbo files

In some targets there are up towards 2000 files (`content/browser`, outside Blink) so there is a question of whether putting that many files together is a win. Experiments show that the more files you group together, the fewer CPU cycles are used, but the parallelity also drops which can result in an increase in total build time.

For a 4 cores/8 threads machine compiling `content` (not Blink) in a jumbo build gave only small increases in build time past 200 cc files per jumbo file.

Experiment

Content with renderer, browser, utility and child using unity and common being the normal system (which adds a base level of a couple of minutes):

Max size of jumbo file	**Total targets**	**Wall time (-j 3)**	**CPU time**
Part of time below used for non-jumbo targets	*182*	*0m55s*	*3m30s*
Disabled jumbo	1331	13m56s	41m37s
5	736	9m22s	28m01s
10	489	5m36s	16m45s
50	286	4m29s	13m21s
100	262	4m00s	11m58s
200	**248**	**3m43s**	**10m53s**
400	244	3m41s	10m13s
600	243	3m31s	9m42s
800	237	4m07s	9m38s
2000	236	4m05s	9m45s

Making the jumbo files larger than necessary just adds risk of breaking some tool so this seems to indicate that a suitable max size is in the range of 100-600 files.

Single-file-editing scenario

The current version of the system makes it possible to exclude one target from jumbo so that it's possible to edit and recompile a single cpp file as quickly as before. I am not sure this will be an overall win for the developer since the time gained will be lost whenever a header file is changed, but it is a possibility to have this feature.

Special cases

For files that are not suitable to combine with others, there is a "`jumbo_excluded_sources`" list that can be used.
Targets that should not be jumbo compiled at all can either avoid using jumbo_target or set `jumbo_build_override=false`
Targets that should always be jumbo compiled can set `jumbo_build_override=true`

Configuration

A global gn flag `use_jumbo_build` is set to `true` or `false` (default `true` for most configurations) to control if unity builds are used for configured targets.
A global string `jumbo_build_excluded` can be set to a target name to exclude that target from unity builds (see "Single-file-editing scenario" above).

Implementation

~~Phase 1: Get it running for blink core. Measure results.~~
Phase 2: If results are good. Post public intention.
Phase 3: If the response is good, start landing patches. (If the response is bad, ~~land internally in Opera instead?~~)
Phase 4: ???
Phase 5: Profit

Patch steps:

1. ~~Base scripts~~
2. ~~Add to for blink core and blink modules~~
3. ~~Land patches that remove jumbo exceptions in blink core and blink modules~~
4. Make it supported by one or more compilation bots (make it default on?)
5. Enable for content
    a. IPC headers
    b. X11 headers
    c. Name collisions
    d. BUILD.gn
6. Land patches that remove jumbo exceptions in content
7. Expand to blink/platform and blink/web and blink/unittests
8. Expand to //ui and //net
9. Use in Opera UI code
10. Find someone to apply this Chrome specific code?

IPC headers?

Remaining questions

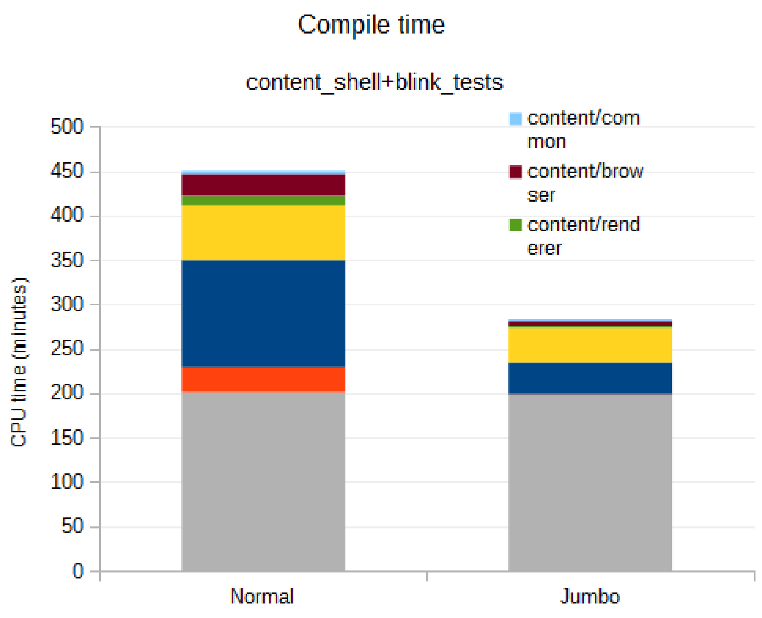
Only for debug? Only for non-official?
In only for debug, commitqueue won't be that much faster.

Results

Converting just part of blink and part of content is enough to show a big difference. Normally a release build of content_shell + blink_tests is 451 CPU minutes. With the limited unity build it goes down to 284 minutes for a saving of
166 CPU minutes or -37%.

The changes are distributed as:
-86 CPU minutes blink/core
-26 CPU minutes blink/bindings
-21 CPU minutes in blink/modules
-19 CPU minutes in content/browser
-9 CPU minutes in content/renderer
-2 CPU minutes in content/common
See Appendix A for raw data

A side benefit is that the disk usage for object files shrinks by 70-80%. It is possible the disk requirements can be lowered somewhat.



goma

goma is a Google-internal distributed compilation system which greatly assists in build times for those that can use it.

Testing indicate that jumbo speeds up many goma builds as well. Since goma acts as a massively parallel build machine it is not obvious that goma assisted builds would benefit from jumbo, but the data indicates that full rebuilds with just Blink core using jumbo can be 5-15% faster.

The breakeven point between when it is slightly slower (single cpp file edits) and when it is faster is different with goma. On an ordinary computer it is enough to have to recompile a few dozen file to benefit from jumbo. With goma you might not gain much unless you have to recompile several times more.

Suggested coding style changes

Namespaces

Use custom namespaces named from the file for unittest or browsertest files that don't export any symbols for use by others. This makes it possible to copy/paste code between test files without getting clashes in jumbo builds.

Example foo_something_unittest.cc:
```
Namespace content {
Namespace foo_something_unittest {
// Help functions and classes
…
// Tests
...
} // namespace foo_something_unittest
} // namespace content
```

This has been used in some parts of Chromium but its invasiveness means that it's not a popular solution. Seek wide code owner support before applying it.

Naming

Make all symbol names unique in the build target. This avoids jumbo compilation problems and make it easier to grep, debug and understand symbols in stack traces or program analysis.

Polymorphic headers controlled by defines

If a header can be different depending on defines, that define must have the same value in the whole build target. Otherwise that header can be different in jumbo builds than in non-jumbo builds.

Bad:

Foo.h
```
class Foo() {
```

```
static int GetValue() {
#ifdef FOO_HIGH_VALUE
return 100;
#else
return 2;
#endif
}
```

Foo.cpp
```
#define FOO_HIGH_VALUE
#include "Foo.h"
const kConstant = Foo::GetValue();  // Assumes 100
```

Bar.cpp
```
#include Foo.h
const kConstant = Foo::GetValue(); // Assumes 2
```

Better is to have different methods, or worst case, split the different interpretations into different build targets so that the build system can decide whether to set FOO_HIGH_VALUE or not.

Note that the code above is also illegal by the C++ One Definition Rule unless GetValue() always get inlined.

General advice

Don't copy even small functions. Instead find a common location for the source. This is sometimes hard in the fractured code base that is Chromium, but too much code duplication has brought it to a bad place and we should stop making it worse. Also, code that is copied tend to collide in jumbo builds.

Pitfalls

Runtime errors

Silent use of unintended variable

This can happen if the intended variable is declared in a scope further away than another scope with the same symbol.

Example (pseudo code):

```
FileA.cpp
namespace content { const int kConstant = 10; }
FileB.cpp
namespace { const int kConstant = 10000; }
namespace content { printf("%d\n", kConstant); }
```

This will print 10 if FileA.cpp and FileB.cpp are compiled together, 10000 otherwise.

Risk

There is a non-negligible risk that there are such conflicts where the colliding variable has the same name, the same type, but a different value and then the code will compile with no error or warning.

I have noticed 1 near-instance of this in `content` (same name, different type). Might be 0 or many cases undetected. For that reason alone I'm reluctant to recommend this for official builds initially.

Compiler errors

Conflicting/reused symbol names.

Easy to avoid by making names unique or reducing the scope of the symbol name or, if the symbols refer to identical functions/data, share the data.

Leaking #defines

Includes that leak macros will affect many more files and can cause strange errors.
FOURCC is used internally in both libyuv and webrtc

Suggested fix is to wrap the includes of those third party libraries in an include that undefs the offending symbols and that is used everywhere.

X11 defines a lot of True, None, False, … and similar symbols often used for constants or enums elsewhere..
Fix: Define the constants in an X11Constants namespace/class/struct after undefining them.

Leaking #undefs

Only encountered one of these, an attempt to prevent a macro from being used in a file.
Can be solved with #pragma push_macro and #pragma pop_macro or by not doing the #undef.

Missing includes

With a jumbo build you get includes from other .cc files and that might make you forget to do the includes locally. This will then break non-jumbo builds.

Compiler warnings only for headers (Android+GCC)

GCC warns (which turns into an error through -Werror) about certain things for headers which it ignores for cpp files. Like

```
FAILED: obj/third_party/WebKit/Source/core/animation/animation/animation_jumbo_1.o
In file included from gen/third_party/WebKit/Source/core/animation/animation_jumbo_1.cc:15:0:
gen/third_party/WebKit/Source/core/animation/../../../../../../../../third_party/WebKit/Source
/core/animation/CSSBorderImageLengthBoxInterpolationType.cpp:76:7: error:
'blink::CSSBorderImageLengthBoxNonInterpolableValue' has a field
'blink::CSSBorderImageLengthBoxNonInterpolableValue::side_types_' whose type uses the
anonymous namespace [-Werror]
 class CSSBorderImageLengthBoxNonInterpolableValue
```

Duplicate symbols in the linker

When the GNU linker considers symbols to include in a binary from an `.a` file, it does so one `.o` file at time. If there are duplicate symbols in two `.o` files and one of them is not needed then there will be no linker error. If the two `.o` files are merged with other .o files which are needed, you get linker errors about duplicate symbols.

Typically easily fixed by just not having the same code compiled twice.

Duplicate template definitions in the linker

Windows Debug, component build only.
If a class inherits publicly from a template and exports it implicitly (i.e. MODULES_EXPORT in modules) and the same compilation unit is also importing the identical template by using another class that inherits publicly from the same template (i.e. CORE_EXPORT in modules), then Visual Studio 2015 will fail with a linker error:

```
FAILED: blink_modules.dll blink_modules.dll.lib
c:/Program Files/Python27/python.exe ../../build/toolchain/win/tool_wrapper.py link-wrapper
environment.x64 False link.exe /nologo /IMPLIB:./blink_modules.dll.lib /DLL
/OUT:./blink_modules.dll /PDB:./blink_modules.dll.pdb @./blink_modules.dll.rsp
blink_core.dll.lib(blink_core.dll) : error LNK2005: "public: __cdecl blink::PairIterable<class
WTF::String,class WTF::String>::PairIterable<class WTF::String,class WTF::String>(void)"
(??0?$PairIterable@VString@WTF@@V12@@blink@@QEAA@XZ) already defined in FetchEventInit.obj
./blink_modules.dll : fatal error LNK1169: one or more multiply defined symbols found
```

Can be fixed by making sure that the import and export is not in the same compilation unit with a (Windows only?) `jumbo_excluded_sources`.

Headers that change based on previously defined macros

V8’s src/arm64/instructions-arm64.h produces two different results, depending on whether or not ARM64_DEFINE_FP_STATICS is defined, and required that it be included exactly once with that macro defined. Since it has a header guard, jumbo compilation units would only use the first instance to appear, which could result in either multiply defined or undefined symbols in some (many) cases.

Fix in progress: https://chromium-review.googlesource.com/c/v8/v8/+/663898

Visual Studio / IDE integration

With jumbo foo.cc files look like headers to gn and IDEs. That may make them less well supported. For instance Ctrl+F7 in Visual Studio to compile just a single cc file will not work if that cc file has been merged into a jumbo unit.

This could probably be improved if/when jumbo becomes a native gn feature. Until then the best workaround is to disable jumbo for the active target by setting `"jumbo_build_excluded=mytarget"`.

Appendix A - compile times

With Jumbo on some of blink and some of content

Compile times content_shell blink_tests with jumbo enabled -j 3, based on bc7560c7f88ce2ab4ea1f6389d79084cfa2d1c34

Total 284 CPU minutes.39 CPU minutes in blink/modules, 35 in blink/core, 24 in blink/platform (not converted), 21 in v8 (not converted), 18 in blink/unittests (not converted), 18 in ui (not converted), 17 in content, 16 in blink/web (not converted), 13 in net (not converted), 10 in webrtc (not converted), 10 in cc (not converted), 8 in skia (not converted)

(5 in content/browser, 2 content/common, 2 content/renderer, 1.5 in blink/bindings)

CPU minutes, percentage of all, target path

```
283.996     100    /
279.512    98.42   //obj
143.622    50.57   //obj/third_party
117.755    41.46   //obj/third_party/WebKit
115.484    40.66   //obj/third_party/WebKit/Source
```

39.476	13.9	//obj/third_party/WebKit/Source/modules
35.161	12.38	//obj/third_party/WebKit/Source/core
27.944	9.84	//obj/third_party/WebKit/Source/modules/modules
23.344	8.22	//obj/third_party/WebKit/Source/platform
21.429	7.545	//obj/v8
18.569	6.539	//obj/v8/v8_base
18.118	6.38	//obj/third_party/WebKit/Source/core/unit_tests
17.709	6.236	//obj/ui
17.519	6.169	//obj/content
15.974	5.625	//obj/third_party/WebKit/Source/web
12.693	4.469	//obj/net
12.578	4.429	//obj/third_party/WebKit/Source/platform/platform
11.063	3.895	//obj/net/net
9.875	3.477	//obj/third_party/webrtc
9.675	3.407	//obj/cc
8.094	2.85	//obj/third_party/WebKit/Source/web/web
7.999	2.817	//obj/skia
7.842	2.761	//obj/third_party/WebKit/Source/web/webkit_unit_tests
7.658	2.696	//obj/skia/skia
7.438	2.619	//obj/services
6.974	2.456	//obj/media
6.327	2.228	//obj/ppapi
6.237	2.196	//obj/ui/views
5.908	2.08	//obj/third_party/WebKit/Source/platform/blink_platform_unittests
5.159	1.816	//obj/device
5.019	1.767	//obj/content/browser
4.991	1.758	//obj/third_party/WebKit/Source/core/core_generated
4.877	1.717	//obj/content/browser/browser
4.835	1.702	//obj/third_party/webrtc/modules
4.832	1.702	//obj/ui/views/views
4.633	1.631	//obj/cc/cc
4.582	1.614	//obj/third_party/WebKit/Source/modules/unit_tests
4.455	1.569	//obj/mojo
4.445	1.565	//obj/base
4.147	1.46	//obj/components
4.032	1.42	//gen
4.023	1.416	//obj/content/test
3.948	1.39	//obj/services/ui
3.842	1.353	//obj/gpu
3.803	1.339	//obj/content/test/test_support
3.687	1.298	//obj/third_party/angle
3.673	1.293	//obj/ppapi/proxy
3.286	1.157	//obj/base/base

3.174	1.118	//obj/ppapi/proxy/proxy
2.987	1.052	//obj/mojo/public
2.815	0.9912	//obj/content/shell
2.724	0.9592	//obj/third_party/protobuf
2.440	0.8592	//obj/ui/aura
2.352	0.8282	//obj/device/bluetooth
2.302	0.8104	//obj/mojo/public/interfaces
2.302	0.8104	//obj/mojo/public/interfaces/bindings
2.300	0.8098	//obj/mojo/public/interfaces/bindings/tests
2.271	0.7996	//obj/third_party/WebKit/public
2.266	0.798	//obj/content/common
2.092	0.7367	//obj/cc/ipc
2.084	0.7337	//gen/blink
2.046	0.7203	//obj/third_party/angle/libANGLE
2.035	0.7165	//obj/gpu/command_buffer
1.975	0.6953	//obj/content/renderer
1.962	0.6908	//obj/third_party/WebKit/Source/platform/wtf
1.891	0.666	//obj/ui/gfx
1.862	0.6558	//obj/content/renderer/renderer
1.839	0.6476	//gen/blink/bindings
1.773	0.6242	//obj/storage
1.755	0.618	//obj/ui/gl
1.741	0.6132	//obj/device/bluetooth/bluetooth
1.741	0.6131	//obj/content/shell/content_shell_lib
1.731	0.6096	//obj/ui/aura/aura
1.716	0.6042	//obj/third_party/icu
1.679	0.5912	//obj/storage/browser
1.678	0.5909	//obj/storage/browser/browser
1.677	0.5906	//obj/services/ui/ws
1.630	0.5738	//obj/services/ui/ws/lib
1.615	0.5686	//obj/cc/test_support
1.543	0.5435	//obj/v8/v8_builtins_generators
1.535	0.5405	//obj/gpu/command_buffer/service
1.526	0.5372	//obj/third_party/WebKit/Source/bindings
1.504	0.5296	//obj/gpu/command_buffer/service/service_sources
1.440	0.5069	//obj/third_party/WebKit/Source/core/inspector
1.439	0.5068	//obj/third_party/mesa
1.425	0.5019	//obj/services/ui/public
1.424	0.5016	//obj/third_party/WebKit/Source/core/layout
1.424	0.5015	//obj/media/media
1.401	0.4934	//obj/third_party/WebKit/Source/bindings/core
1.400	0.4931	//obj/third_party/WebKit/Source/bindings/core/v8
1.375	0.4841	//obj/gpu/ipc

```
  1.363          0.4801 //obj/ppapi/thunk
  1.363          0.48    //obj/ppapi/thunk/thunk
  1.362          0.4794 //obj/third_party/protobuf/protoc_lib
  1.346          0.4739 //obj/media/mojo
  1.257          0.4427 //obj/third_party/WebKit/Source/platform/wtf/wtf_unittests
  1.238          0.436   //obj/third_party/webrtc/pc
  1.226          0.4317 //obj/ui/events
  1.217          0.4286 //obj/third_party/WebKit/Source/core/layout/layout
  1.217          0.4285 //obj/third_party/WebKit/Source/core/html
  1.216          0.4281 //obj/third_party/WebKit/Source/core/html/html
  1.207          0.4251 //obj/net/test_support
  1.207          0.425   //obj/services/ui/public/interfaces
  1.205          0.4242 //obj/services/service_manager
  1.194          0.4205 //obj/third_party/webrtc/modules/audio_coding
  1.180          0.4154 //obj/third_party/WebKit/Source/bindings/core/v8/bindings_core_impl
  1.177          0.4143 //obj/third_party/protobuf/protobuf_full
  1.131          0.3984 //obj/third_party/WebKit/Source/core/editing
  1.130          0.3981 //obj/ui/base
  1.121          0.3947 //obj/third_party/angle/translator
  1.1130.3918 //obj/third_party/mesa/mesa
  1.1120.3916 //obj/third_party/WebKit/Source/platform/loader
  1.098          0.3867 //obj/third_party/webrtc/modules/audio_processing
  1.086          0.3826 //obj/media/base
  1.078          0.3795 //obj/third_party/WebKit/Source/core/dom
  1.077          0.3792 //obj/third_party/WebKit/Source/core/dom/dom
  1.062          0.3739 //obj/media/base/base
  1.054          0.371   //obj/third_party/webrtc/modules/audio_processing/audio_processing
  1.052          0.3706 //obj/third_party/libvpx
  1.032          0.3636 //obj/third_party/icu/icui18n
  1.027          0.3616
//obj/third_party/WebKit/Source/bindings/core/v8/bindings_core_impl/bindings_core_impl_jumbo
_2.o
  1.003          0.3533 //obj/content/common/mojo_bindings
  0.991          0.3488 //obj/services/service_manager/public
  0.988          0.348   //obj/third_party/WebKit/Source/core/css
  0.987          0.3477 //obj/third_party/WebKit/Source/core/css/css
  0.972          0.3423 //obj/content/shell/test_runner
  0.971          0.3418 //obj/content/shell/test_runner/test_runner
  0.940          0.331   //gen/blink/bindings/modules
  0.929          0.3272 //obj/v8/src
  0.929          0.3272 //obj/v8/src/inspector
  0.928          0.3267 //obj/v8/src/inspector/inspector
  0.921          0.3242 //gen/blink/bindings/modules/v8
```

0.911	0.3209	//obj/ui/views/test_support_internal
0.905	0.3188	//obj/third_party/webrtc/modules/rtp_rtcp
0.905	0.3186	//obj/third_party/webrtc/modules/rtp_rtcp/rtp_rtcp
0.900	0.317	//obj/cc/ipc/interfaces
0.897	0.3157	//gen/blink/bindings/core
0.897	0.3157	//obj/third_party/webrtc/pc/libjingle_peerconnection
0.897	0.3157	//obj/components/metrics
0.882	0.3107	//obj/third_party/WebKit/Source/core/inspector/generated
0.879	0.3096	//gen/blink/bindings/core/v8
0.878	0.3091	//obj/ui/compositor
0.871	0.3067	//obj/content/browser/browser/browser_jumbo_3.o
0.843	0.2968	//obj/gpu/ipc/common
0.842	0.2964	//obj/ui/gl/gl_unittest_utils
0.835	0.2941	//obj/content/browser/browser/browser_jumbo_1.o
0.834	0.2937	//obj/content/browser/browser/browser_jumbo_6.o
0.821	0.289	//obj/ui/gl/gl
0.803	0.2826	//obj/third_party/WebKit/Source/platform/heap
0.793	0.2793	//obj/third_party/webrtc/base
0.787	0.2772	//obj/mojo/edk
0.783	0.2757	//obj/base/test
0.779	0.2744	//obj/mojo/public/interfaces/bindings/tests/test_interfaces_blink
0.769	0.2709	//obj/base/test/test_support
0.766	0.2697	//obj/content/common/common
0.755	0.266	//obj/ui/base/base
0.753	0.2652	//obj/media/capture
0.749	0.2639	//obj/media/mojo/interfaces
0.734	0.2584	//obj/cc/ipc/interfaces_blink
0.719	0.2531	//obj/third_party/WebKit/public/mojo_bindings_blink
0.717	0.2526	//obj/media/audio
0.717	0.2523	//obj/media/audio/audio
0.708	0.2492	//obj/ui/aura/test_support
0.704	0.2479	//obj/third_party/WebKit/Source/platform/wtf/platform_wtf
0.697	0.2456	//obj/services/service_manager/public/interfaces
0.696	0.2451	//obj/content/browser/browser/browser_jumbo_2.o
0.696	0.245	//obj/mojo/public/interfaces/bindings/tests/test_interfaces
0.685	0.2413	//obj/third_party/ffmpeg
0.682	0.2403	//obj/mojo/public/cpp
0.682	0.2402	//obj/third_party/icu/icuuc
0.681	0.2398	//obj/ui/gl/gl_unittest_utils/gl_mock.o
0.680	0.2395	//obj/mojo/common
0.676	0.2382	//obj/third_party/webrtc/modules/video_coding
0.672	0.2365	//obj/third_party/WebKit/public/mojo_bindings
0.667	0.2348	//obj/ui/gfx/gfx

```
  0.666           0.2344 //obj/breakpad
  0.660           0.2325 //obj/ppapi/shared_impl
  0.653           0.2299 //obj/ipc
  0.651           0.2292 //obj/content/browser/browser/browser_jumbo_5.o
  0.647           0.2279 //obj/content/renderer/renderer/renderer_jumbo_2.o
  0.626           0.2205 //obj/ppapi/shared_impl/shared_impl
  0.620           0.2182 //obj/mojo/public/cpp/bindings
  0.612           0.2154 //obj/ui/wm
  0.605           0.2129 //obj/third_party/WebKit/Source/core/editing/editing
  0.597           0.2102 //obj/third_party/webrtc/media
  0.597           0.2102 //obj/third_party/libvpx/libvpx
  0.592           0.2083 //obj/third_party/WebKit/Source/core/layout/layout/layout_jumbo_1.o
  0.590           0.2076 //obj/ui/display
  0.576           0.2027 //obj/google_apis
  0.575           0.2025 //obj/google_apis/google_apis
  0.569           0.2003 //obj/third_party/webrtc/p2p
  0.569           0.2003 //obj/ui/wm/wm
  0.565           0.1989 //obj/third_party/boringssl
  0.561           0.1975 //obj/third_party/WebKit/Source/platform/test_support
  0.559           0.1967 //obj/device/usb
  0.555           0.1954
//obj/third_party/WebKit/Source/core/inspector/inspector/inspector_jumbo_1.o
  0.555           0.1954 //obj/third_party/WebKit/Source/core/inspector/inspector
  0.554           0.1952 //obj/third_party/boringssl/boringssl
  0.554           0.195   //obj/services/ui/public/interfaces/interfaces
  0.553           0.1947 //obj/third_party/WebKit/Source/platform/loader/loader
```

Without jumbo at all

Compile times content_shell blink_tests -j 3, based on
bc7560c7f88ce2ab4ea1f6389d79084cfa2d1c34

Total 451 CPU minutes. 121 in blink/core 61 CPU minutes in blink/modules, 28 in blink/bindings, 24 in content/browser, 23 in blink/platform, 22 in v8, 18 in blink/unittests, 18 in ui, 16 in blink/web, 13 in net, 11 in content/renderer, 10 in webrtc, 10 in cc, 8 in skia.

CPU minutes, percentage of all, target path

```
 451.143          100    /
 446.705          99.02  //obj
 277.935          61.61  //obj/third_party
 252.112          55.88  //obj/third_party/WebKit
 249.812          55.37  //obj/third_party/WebKit/Source
```

121.091	26.84	//obj/third_party/WebKit/Source/core
61.462	13.62	//obj/third_party/WebKit/Source/modules
50.134	11.11	//obj/content
27.965	6.199	//obj/third_party/WebKit/Source/bindings
27.914	6.187	//obj/third_party/WebKit/Source/modules/modules
27.841	6.171	//obj/third_party/WebKit/Source/bindings/core
27.839	6.171	//obj/third_party/WebKit/Source/bindings/core/v8
27.619	6.122	//obj/third_party/WebKit/Source/bindings/core/v8/bindings_core_impl
24.399	5.408	//obj/content/browser
24.256	5.376	//obj/content/browser/browser
23.324	5.17	//obj/third_party/WebKit/Source/platform
21.501	4.766	//obj/v8
18.629	4.129	//obj/v8/v8_base
18.111	4.014	//obj/third_party/WebKit/Source/core/unit_tests
17.775	3.94	//obj/ui
15.967	3.539	//obj/third_party/WebKit/Source/core/layout
15.966	3.539	//obj/third_party/WebKit/Source/web
12.932	2.866	//obj/third_party/WebKit/Source/core/html
12.712	2.818	//obj/net
12.561	2.784	//obj/third_party/WebKit/Source/platform/platform
11.082	2.456	//obj/content/renderer
11.066	2.453	//obj/net/net
10.973	2.432	//obj/content/renderer/renderer
10.644	2.359	//obj/third_party/WebKit/Source/core/css
9.880	2.19	//obj/third_party/webrtc
9.699	2.15	//obj/cc
8.664	1.921	//obj/third_party/WebKit/Source/core/editing
8.100	1.795	//obj/third_party/WebKit/Source/web/web
8.000	1.773	//obj/skia
7.981	1.769	//obj/third_party/WebKit/Source/core/dom
7.827	1.735	//obj/third_party/WebKit/Source/web/webkit_unit_tests
7.655	1.697	//obj/skia/skia
7.462	1.654	//obj/services
7.070	1.567	//obj/third_party/WebKit/Source/core/svg
7.069	1.567	//obj/third_party/WebKit/Source/core/svg/svg
6.979	1.547	//obj/media
6.931	1.536	//obj/third_party/WebKit/Source/core/paint
6.340	1.405	//obj/ppapi
6.272	1.39	//obj/ui/views
6.168	1.367	//obj/third_party/WebKit/Source/core/animation
5.908	1.309	//obj/third_party/WebKit/Source/platform/blink_platform_unittests
5.182	1.149	//obj/device
4.970	1.102	//obj/third_party/WebKit/Source/core/core_generated

4.858	1.077	//obj/ui/views/views
4.845	1.074	//obj/third_party/webrtc/modules
4.654	1.032	//obj/cc/cc
4.565	1.012	//obj/third_party/WebKit/Source/modules/unit_tests
4.460	0.9886	//obj/base
4.451	0.9865	//obj/mojo
4.169	0.9241	//obj/components
4.130	0.9154	//obj/third_party/WebKit/Source/modules/webgl
4.129	0.9152	//obj/third_party/WebKit/Source/modules/webgl/webgl
4.036	0.8946	//obj/content/test
3.968	0.8796	//gen
3.962	0.8782	//obj/services/ui
3.836	0.8503	//obj/gpu
3.816	0.8459	//obj/content/test/test_support
3.685	0.8168	//obj/third_party/angle
3.681	0.8159	//obj/ppapi/proxy
3.514	0.7788	//obj/third_party/WebKit/Source/core/paint/paint_1
3.503	0.7765	//obj/third_party/WebKit/Source/core/layout/layout_1
3.416	0.7572	//obj/third_party/WebKit/Source/core/paint/paint_0
3.413	0.7565	//obj/content/common
3.340	0.7404	//obj/third_party/WebKit/Source/core/layout/layout_0
3.294	0.7302	//obj/base/base
3.284	0.728	//obj/third_party/WebKit/Source/core/inspector
3.181	0.705	//obj/ppapi/proxy/proxy
2.985	0.6617	//obj/third_party/WebKit/Source/core/html/html_1
2.981	0.6607	//obj/mojo/public
2.956	0.6551	//obj/third_party/WebKit/Source/core/layout/svg
2.955	0.655	//obj/third_party/WebKit/Source/core/layout/svg/svg_layout
2.943	0.6523	//obj/third_party/WebKit/Source/core/html/html_0
2.812	0.6234	//obj/third_party/WebKit/Source/core/editing/unit_tests
2.806	0.6219	//obj/content/shell
2.791	0.6186	//obj/third_party/WebKit/Source/core/frame
2.790	0.6185	//obj/third_party/WebKit/Source/core/frame/frame
2.727	0.6045	//obj/third_party/protobuf
2.699	0.5982	//obj/third_party/WebKit/Source/core/css/css_4
2.483	0.5504	//obj/third_party/WebKit/Source/core/html/html_2
2.451	0.5433	//obj/ui/aura
2.418	0.5359	//obj/third_party/WebKit/Source/core/inspector/inspector
2.406	0.5334	//obj/third_party/WebKit/Source/core/html/html_4
2.368	0.525	//obj/device/bluetooth
2.348	0.5206	//obj/third_party/WebKit/Source/core/layout/layout_3
2.300	0.5099	//obj/third_party/WebKit/public
2.294	0.5085	//obj/mojo/public/interfaces

2.294	0.5085	//obj/mojo/public/interfaces/bindings
2.292	0.5081	//obj/mojo/public/interfaces/bindings/tests
2.192	0.4858	//obj/third_party/WebKit/Source/core/layout/layout_4
2.180	0.4832	//obj/third_party/WebKit/Source/core/loader
2.179	0.4831	//obj/third_party/WebKit/Source/core/loader/loader
2.155	0.4778	//obj/third_party/WebKit/Source/core/events
2.155	0.4776	//obj/third_party/WebKit/Source/core/events/events
2.122	0.4704	//obj/third_party/WebKit/Source/modules/accessibility
2.122	0.4703	//obj/third_party/WebKit/Source/modules/accessibility/accessibility
2.112	0.468	//obj/third_party/WebKit/Source/core/css/css_1
2.110	0.4676	//obj/third_party/WebKit/Source/core/html/html_3
2.086	0.4624	//obj/cc/ipc
2.064	0.4575	//obj/third_party/WebKit/Source/modules/webaudio
2.063	0.4574	//obj/third_party/WebKit/Source/modules/webaudio/webaudio
2.059	0.4565	//obj/third_party/WebKit/Source/core/page
2.059	0.4563	//obj/third_party/WebKit/Source/core/page/page
2.051	0.4546	//obj/third_party/angle/libANGLE
2.032	0.4504	//obj/gpu/command_buffer
2.018	0.4473	//gen/blink
2.004	0.4442	//obj/third_party/WebKit/Source/core/css/css_3
1.964	0.4352	//obj/third_party/WebKit/Source/platform/wtf
1.948	0.4318	//obj/third_party/WebKit/Source/core/css/css_2
1.929	0.4275	//obj/content/child
1.928	0.4273	//obj/content/child/child
1.918	0.4251	//obj/content/common/common
1.912	0.4238	//obj/ui/gfx
1.878	0.4162	//obj/third_party/WebKit/Source/core/css/css_0
1.820	0.4034	//obj/content/public
1.816	0.4026	//gen/blink/bindings
1.769	0.392	//obj/storage
1.765	0.3913	//obj/ui/gl
1.758	0.3897	//obj/device/bluetooth/bluetooth
1.747	0.3873	//obj/third_party/WebKit/Source/core/dom/dom_1
1.745	0.3868	//obj/ui/aura/aura
1.731	0.3836	//obj/content/shell/content_shell_lib
1.727	0.3827	//obj/third_party/WebKit/Source/core/animation/animation_0
1.711	0.3792	//obj/third_party/icu
1.682	0.3728	//obj/services/ui/ws
1.675	0.3714	//obj/storage/browser
1.675	0.3712	//obj/storage/browser/browser
1.655	0.3668	//obj/third_party/WebKit/Source/core/dom/dom_3
1.634	0.3622	//obj/services/ui/ws/lib
1.624	0.36	//obj/third_party/WebKit/Source/core/layout/layout_2

1.621	0.3593	//obj/cc/test_support
1.616	0.3582	//obj/third_party/WebKit/Source/core/style
1.601	0.355	//obj/third_party/WebKit/Source/core/dom/dom_2
1.575	0.3491	//obj/third_party/WebKit/Source/core/workers
1.574	0.3489	//obj/third_party/WebKit/Source/core/workers/workers
1.564	0.3467	//obj/third_party/WebKit/Source/core/testing
1.564	0.3466	//obj/third_party/WebKit/Source/core/style/rendering
1.551	0.3438	//obj/v8/v8_builtins_generators
1.543	0.3419	//obj/third_party/WebKit/Source/core/animation/animation_1
1.537	0.3406	//obj/gpu/command_buffer/service
1.536	0.3405	//obj/third_party/WebKit/Source/core/dom/dom_0
1.505	0.3337	//obj/gpu/command_buffer/service/service_sources
1.438	0.3187	//obj/third_party/WebKit/Source/core/dom/dom_4
1.430	0.317	//obj/third_party/mesa
1.429	0.3167	//obj/services/ui/public
1.416	0.3139	//obj/media/media
1.405	0.3114	//obj/third_party/WebKit/Source/core/editing/editing_0
1.371	0.304	//obj/ppapi/thunk
1.371	0.3039	//obj/ppapi/thunk/thunk
1.371	0.3039	//obj/gpu/ipc
1.365	0.3027	//obj/media/mojo
1.361	0.3016	//obj/third_party/protobuf/protoc_lib
1.292	0.2863	//obj/third_party/WebKit/Source/core/editing/editing_1
1.259	0.279	//obj/third_party/WebKit/Source/platform/wtf/wtf_unittests
1.250	0.2772	//obj/third_party/WebKit/Source/core/editing/editing_2
1.245	0.2759	//obj/third_party/WebKit/Source/core/animation/animation_3
1.241	0.2751	//obj/third_party/webrtc/pc
1.227	0.2719	//obj/ui/events
1.211	0.2685	//obj/services/ui/public/interfaces
1.209	0.2679	//obj/net/test_support
1.207	0.2675	//obj/third_party/WebKit/Source/modules/indexeddb
1.206	0.2674	//obj/third_party/WebKit/Source/modules/indexeddb/indexeddb
1.196	0.2652	//obj/services/service_manager
1.191	0.2639	//obj/third_party/webrtc/modules/audio_coding
1.180	0.2615	//obj/third_party/protobuf/protobuf_full
1.124	0.2491	//obj/ui/base
1.118	0.2479	//obj/third_party/angle/translator
1.111	0.2463	//obj/third_party/WebKit/Source/platform/loader
1.107	0.2453	//obj/third_party/mesa/mesa
1.096	0.243	//obj/third_party/webrtc/modules/audio_processing
1.094	0.2426	//obj/third_party/WebKit/Source/modules/serviceworkers
1.094	0.2424	//obj/third_party/WebKit/Source/modules/serviceworkers/serviceworkers
1.083	0.24	//obj/media/base

1.070	0.2372	//obj/third_party/WebKit/Source/core/xml
1.069	0.237	//obj/third_party/WebKit/Source/core/xml/xml
1.058	0.2346	//obj/media/base/base
1.050	0.2328	//obj/third_party/webrtc/modules/audio_processing/audio_processing
1.045	0.2316	//obj/third_party/libvpx
1.033	0.229	//obj/third_party/icu/icui18n
1.018	0.2256	//obj/content/public/browser
1.017	0.2255	//obj/content/public/browser/browser_sources
0.998	0.2211	//obj/content/common/mojo_bindings
0.985	0.2184	//obj/services/service_manager/public
0.981	0.2174	//obj/third_party/WebKit/Source/modules/webdatabase
0.980	0.2172	//obj/third_party/WebKit/Source/modules/webdatabase/webdatabase
0.973	0.2156	//obj/content/shell/test_runner
0.971	0.2153	//obj/content/shell/test_runner/test_runner
0.967	0.2142	//obj/third_party/WebKit/Source/core/editing/editing_3
0.957	0.212	//obj/third_party/WebKit/Source/core/animation/animation_2
0.934	0.2071	//obj/third_party/WebKit/Source/core/editing/editing_4
0.929	0.2058	//obj/v8/src/inspector
0.929	0.2058	//obj/v8/src
0.927	0.2054	//obj/v8/src/inspector/inspector
0.926	0.2053	//gen/blink/bindings/modules
0.920	0.204	//obj/third_party/webrtc/modules/rtp_rtcp
0.920	0.2039	//obj/third_party/webrtc/modules/rtp_rtcp/rtp_rtcp

Appendix B

Results from just using jumbo on Blink core in goma.

```
With jumbo (component debug build + goma -j400)
--------------------------------
full clean build
14m16.920s
15m43.783s
14m59.039s

incremental build after touching LayoutObject.cpp
0m56.785s
0m57.414s
0m57.419s
0m57.515s
0m58.475s
0m55.965s
```

```
incremental build after touching Document.cpp
0m54.941s
0m56.032s
0m55.063s
0m55.974s
0m54.226s
0m54.372s

incremental build after touching Document.h
0m59.336s
1m20.426s
0m20.763s
0m22.330s
0m20.590s
0m20.496s

Without jumbo (component debug build + goma -j400)
--------------------------------
full clean build
21m14.593s
17m54.998s
17m8.946s

incremental build after touching LayoutObject.cpp
0m31.303s
0m32.626s
0m31.810s
0m31.595s
0m31.460s
0m32.731s

incremental build after touching Document.cpp
0m43.678s
0m43.358s
0m44.996s
0m46.118s
0m43.796s
0m44.275s

incremental build after touching Document.h
4m25.942s
0m44.548s
```

```
0m54.576s
13m8.424s
0m50.529s
12m57.068s
```
(These numbers jumped around a lot. I am unsure why.)