Speed:

why it matters
and why we care

Hans Hagen
ConTEXt Meeting
September 2013
Speed

Pages per minute

What happens

What we can do
• Speed matters in an edit-run-preview cycle although this is mostly perception.
• The nicer the interface, the slower it gets, but you seldom set something up so that it is not much of a burden.
• Everything you provide gets used at some point, also in inefficient ways, so best know your weak spots.
• Lots of local (grouped) tweaks leads to many mechanisms kicking in unseen, grouping matters.
• Wrong use of functionality can have drastic and unexpected speed penalties.
we have speed up the baseline performance (in pages per second) as much as possible

we try to identify and optimize critical routines, both at the TeX and Lua end

of course the used hardware machine and versions of LuaTeX and ConTeXt matter

dorecurse {1000} {test

# pages Jan Apr May Sep

(nuts)

1 2 2 2 4 4
10 15 17 17 36 37
100 90 109 110 237 236
1000 185 234 259 509 512
10000 215 258 289 548 557

< 06/2013, LuaTEX: 0.72+, Dell M90, SSD, 4GB, 2.33 Ghz T7600, Windows 8/32 bit

> 06/2013, LuaTEX: 0.72+, Dell 6700, SSD, 16GB, 2.80 Ghz 3840QM, Windows 8/64 bit
What happens

• load macros and Lua code is loaded from the format
• the system gets initialized, think of fonts and languages
• additional (runtime) files are loaded
• text is typeset and eventually gets passed to the page builder
• pages are packaged, this includes reverting to global document states
• the pdf representation is created
• each of these steps has its bottlenecks
• the more we don, the more Lua gets involved
What we can do

- avoid copying boxes where possible
- only enable initializers and finalizers when functionality is used
- be clever with fonts, in usage as well as in supporting features
- use trial runs in multi-pass mechanisms
- avoid too much macro expansion (only matters for tracing)
- accept that more functionality has a price
- improve the engine and cook up more clever low level code

but

- don't compromise functionality
- avoid too obscure code
- forget about optimization by means of combining functionality