GHC status report

Simon Peyton Jones, Simon Marlow
Microsoft Research

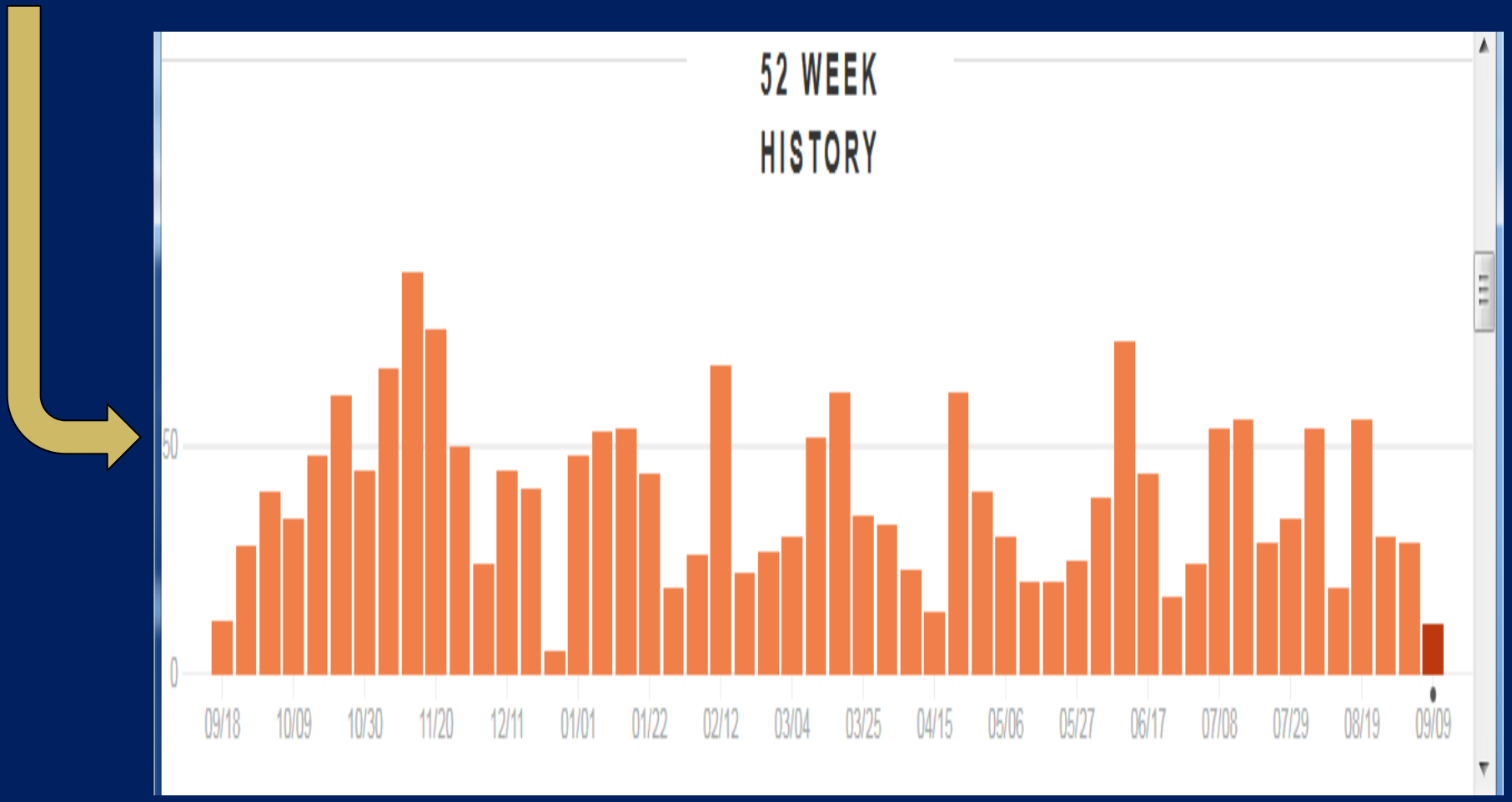
September 2012

What's new in GHC?

- Tons of activity
- Lots of contributors:

  A. Bram Neijt, Austin Seipp, Bas van Dijk, Ben Gamari, David Terei, Dimitrios Vytiniotis, Duncan Coutts, Edward Z. Yang, Eric Kow, Erik de Castro Lopo, Evan Laforge, Gabor Greif, Gabor Pali, Gabriele Keller, Geoffrey Mainland, George Giorgidze, Gregory Wright, "gwern", Ian Lynagh, Iavor S. Diatchki, Joachim Breitner, Joey Adams, Johan Tibell, Jose Pedro Magalhaes, Judah Jacobson, Karel Gardas, Lennart Augustsson, Manuel M T Chakravarty, Max Bolingbroke, Michał Masłowski, Mikhail Vorozhtsov, Mikolaj Konarski, Nathan Howell, Nicolas Trangez, Nils Schweinsberg, Paolo Capriotti, Patrick Palka, Peter Hercek, Peter Wortmann, Reiner Pope, Richard Eisenberg, Ross Paterson, Sam Anklesaria, Samuel Thibault, "shelarcy", Simon Hengel, Simon Marlow, Simon Peyton Jones, Takano Akio, Thomas Dziedzic, Trevor Elliott, Vitaly Bragilesky

50 commits/week



What's new? Syntax!

- \case

```
f :: Maybe Bool -> Int

f (\(Just x) -> x)

f (\case {
      Just x -> x;
      Nothing -> True })
```

Old

New

- multi-if

```
if | x>0        -> e1
   | x==0       -> e2
   | otherwise -> e3
```

What's new? Types!

- Kind polymorphism and data type promotion

```
data T f a = MkT (f a)
-- OLD: T :: (*->*) -> * -> *
-- NEW: T :: forall k. (k->*) -> k -> *
```

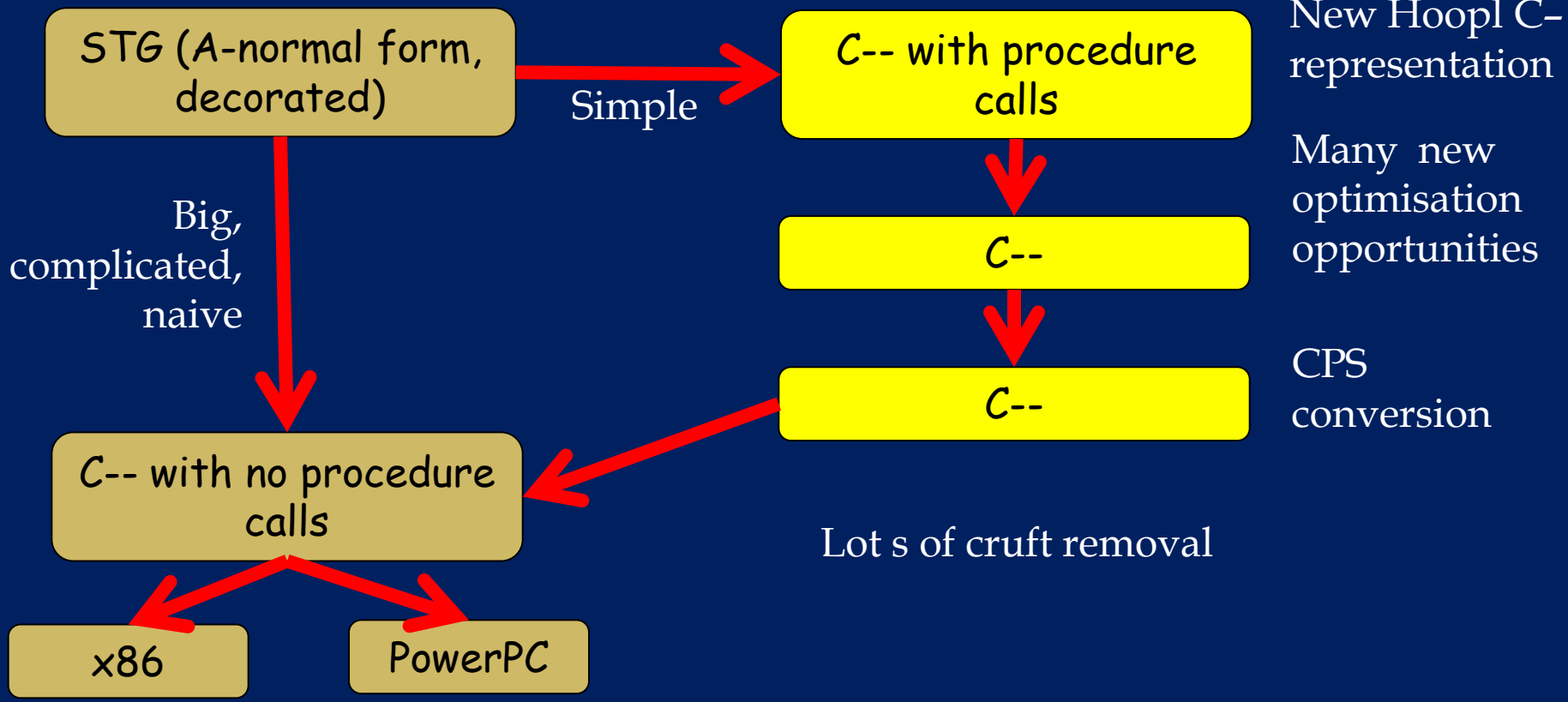
- Deferred type errors
- Holes (demo)
- Type nats (Iavor)

```
append :: Vec n1 -> Vec n2 -> Vec (n1+n2)
```

- Performance improvements (and MUCH more beauty) in the constraint solver

What's new? Code generation

- Long standing project to overhaul the code generator



The "New code generator"

- New optimisation opportunities
  - traditional compiler optimisations have far more opportunity before CPS conversion
  - already better in some cases: one user reported a 2x speedup
  - better calling conventions so we can pass SIMD values in registers
- Will be in 7.8.1
  - but try out a preview in 7.6.1 with "-fnew-codegen"

What's new: platforms

- Win64 version works (thanks IHG)
- Full-scale cross-compilation getting very close

What's new? Parallelism

- Changing number of virtual processors (Simon M)
- SIMD instructions (Geoff)
- Cloud Haskell (Duncan)
- DPH (Ben)

Changing -N at runtime

- (in 7.6.1)
- Previously you had to say +RTS -N2 to use 2 cores, now you can say

```
import GHC.Conc

main = do
  n <- getNumProcessors
  setNumCapabilities n
  ...
```

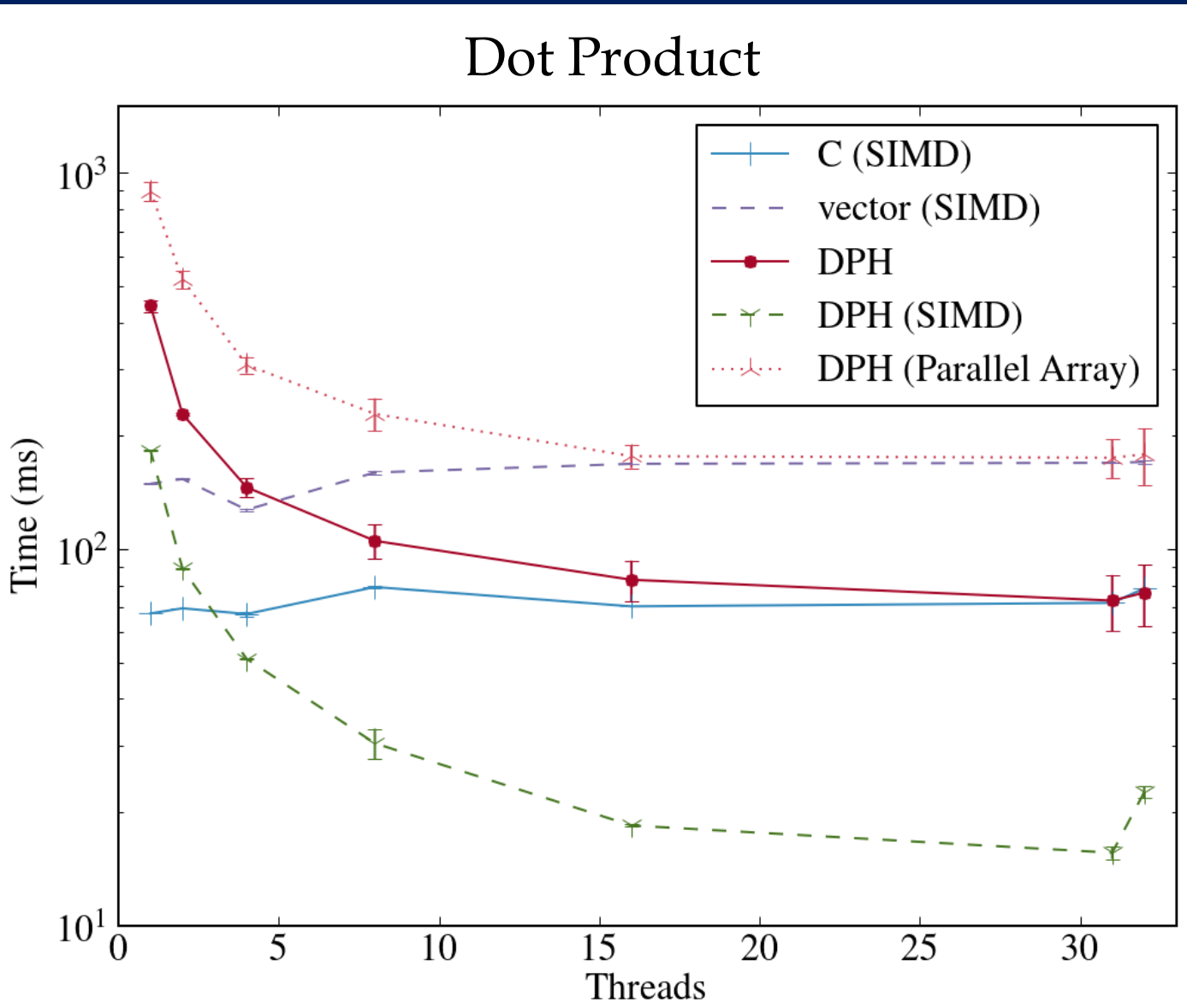
- Call setNumCapabilities any time
- Useful for
  - increasing the parallelism only for the parallel sections of your program
  - no need to tell your users about +RTS -N
  - fork and then setNumCapabilities (for daemons)

- Eventually, this will enable us to dynamically tune the amount of parallelism to respond to contention on the machine
- Diagram shows running a parallel program while periodically changing –N in a cycle 1..8..1..

Status of SIMD Support in GHC

- Low-level primops for SIMD operations.
- Support for stream fusion of SIMD operations in the `vector` library.
- Transparent support for SIMD operations in DPH.

The Good News



The Bad News…

- Register allocation not so hot.
- Not yet in HEAD.

http://hackage.haskell.org/trac/ghc/wiki/SIMD

On the horizon

- Template Haskell overhaul to provide both
  - **Untyped** quotes and splices (for expressions, decls, types, **and patterns**). Allows explicit use of syntax constructors, and analysis of syntax.
  - **Typed** quotes and splices (for expressions only, a la MetaML). Guarantees no type errors in generated code.

```
[e| reverse [True] |]   :: Q Exp

[|| reverse [True] ||]   :: TExp [Bool]
```

On the horizon

- Allow multiple instances of the same package to be installed. For example
  - containers-4.5 built against deepseq-1.4
  - containers-4.5 built against deepseq-1.2
- Needs Cabal changes too
- Avoids some manifestations of package hell

On the horizon

- GHC 7.8 in late 2012 or early 2013
  - New codegen
  - TH overhaul
  - Major DPH release
  - Type holes
  - SIMD

GHC is (still) hot

- Lots going on. Lots. Really a lot.
- We are having way to much fun
- You should join in. We need you.