Haskell as a reagent

Results and Observations on the Use of Haskell in a Python Project

Iustin Pop

Google Inc.

ICFP 2010

Outline

1. Background
   - Our project
   - Haskell additions

2. Observations on the use of Haskell
   - When, how, why

Ganeti

- Cluster-based virtualization management tool
- Started in 2006, open-sourced in 2007
- Used in production:
  - Google internal infrastructure
  - External users (`osuosl.org`, `grnet.net`, etc.)
- Written in Python, uses many opensource libraries/tools

What was missing?

- Sysadmin team, hence focused on “glue” code:
  - system automation
  - integration of tools
- But high performance numerical algorithms were not our focus:
  - Cluster layout was manual, a non-trivial problem (for us)
  - Python-based experiments unsatisfactory

Automatic layout algorithms

- Started with a small idea for a set of bin-packing problems
- First implementation in Python
- Rewrote in OCaml
- Then rewrote in Haskell
- Iterated over many weeks
- Kept the Haskell version
- Ended up with a very dumb automatic cluster layout program

One year and a half later

- Extensive use of the Haskell tools:
  - allocation policy
  - cluster balancing policy
  - node drain (evacuation) policy
  - capacity calculation
- Same core algorithm used in all cases
- Based on multiple cluster metrics
- Extensive integration with the Python code:
  - communication via multiple APIs
  - JSON data format

Outline

1 Background
- Our project
- Haskell additions

2 Observations on the use of Haskell
- When, how, why

Getting a foot in the door

- You must solve a non-trivial problem
- At the right time
- With the right tools
- Unconventional solutions require unconventional problems

Haskell: the good parts

- Solved the problem—in a nice way
  - Good resource usage
  - Nice tools for profiling
  - Easy to extend
- Led to improvements in the Python codebase
  - better API consistency
  - small-scale use of persistent data types

Parameter validation

- Framework developed after the paper was written
- Roughly 100 lines of Python, declaring some base “parameter types”
- Which can be combined to match most usual Python types

Haskell

```
data HVType = HVXen | HVKVM | HVLXC
data EHVMod = Maybe [HVType]
f ::  EHVMod -> Data.Map String Int -> ...
```

Python

```
HVType = TElemOf([‘‘HVXen’’, ‘‘HVKVM’’, ‘‘HVLXC’’])
EHVMod = TOr(TNone, TListOf(HVType))
def fn(x, y):
    EHVMod(x) # raises exception if not correct
    TDictOf(TString, TInt)(y)
```

Haskell: the bad parts

- Production readiness:
  - ease of deployment (good)
  - stability (good)
  - ease of debugging (uh. . .):
    - stack traces
    - logging
    - error handling
- Lazy behaviour (but no so much)
- String issues

Summary

- Haskell can be a viable languange in system administration:
  - High performance, low resource usage, highly expressive
  - It pairs well enough with other languages
- If the problem is well-chosen:
  - should take advantage of language
  - should be non-trivial
- It's fun! *IMHO, YMMV, etc.*

For Further Reading

- `http://code.google.com/p/ganeti`
- `http://git.ganeti.org/`