Experience on implementing a Web server in Haskell

Kazu Yamamoto
Why a New Web Server?

In last fall, I needed a Web server for our research. It should be able to be modified as I want.

Apache is first choice but large and complicated. And I was tired from reading/writing in C.

Yes, I wanted a Web server in Haskell. But I didn't know any Web servers in Haskell.

So, I started programming from scratch. My web server is "Mighttpd" (called mighty)
Three Goals of Mighttpd

- **Functionality**: Mighttpd should provide enough functionality to replace Apache on my domain "Mew.org".

- **Modularity**: Mighttpd should be able to be modified easily for our research.

- **Performance**: Mighttpd should exceed Apache on static contents.
Two Ideas for Performance

 Traditionally, Haskell's `String` type is very slow.

 `ByteString` is faster like `char[]` in C.

 Kernel thread is heavy.

 User thread is light.
HTTP and thread programming

Network protocol
- Message oriented: DNS
- Stream oriented: SMTP, HTTP

Network programming
- Event driven: select, kqueue, epoll
- Threading: fork, pthread_create

🤔 Event driven programming for stream oriented protocol is messy.
😊 Thread programming for stream oriented protocol is concise.
😠 I want to implement HTTP on threading. Simplicity is a good thing.
User Thread is Real Thread

- GHC has an IO manager as a user thread. It is event-driven.
- It takes care of buffering and wakes up blocked user threads.
- So, using user threads is really thread programming.

Diagram:
- hGet
- Handle
- Socket
- User thread
- Buffering
- Event driven
The barrier of 1,024 connections

⚠️ The IO manager is implemented using `select`.

😢 `select` cannot handle over 1,024 files/connections.

☠️ If GHC 6.12 receives over 1,024 connections, resource exhaustion exception happens.

![Diagram with 1024 balloons and select button]
Prefork library

 Prefork is a technique to share a listening port among forked processes.

 ![Prefork diagram](image)

 Kernel

 ![80 connections](image)

 Now, GHC 6.12 can accept any number of connections!
Mighttpd implementation

Package name

- **mighttpd**: File base, KVS base (Not released)
- **webserver**: HTTP, session, redirect, CGI
- **c10k**: prefork

Modularity
"webserver" is designed to handle any storage systems.

Functionality
"mighttpd" works on Mew.org now!
Benchmark Environment

4 JMeters

Controller

JMeter

One static contents
Benchmark Result

- Benchmark is unstable, so don’t fully trust this result.
Profiling

File IO is dominant. Why, Mighttpd slower than Apache?

```
% ab -n 2000 -c 200 -k http://localhost/

COST CENTRE  MODULE  %time  %alloc
fileGet       File    73.3    37.4
mighty        File    20.0    57.9
fileInfo      File    6.7     2.9
fileMapper    File    0.0     1.1
```

Ah, it's overhead of select!

Any hopes?
One Hope

Tibbe and Bos are developing "event" library for kqueue and epoll.

Now we can use it for event-driven network programming.

They are planning to integrate it into the IO manager in GHC 6.14.
Feture architecture

Since there is only one IO manager, GHC 6.14 would not balance on multi-core.

But the prefork technique could be used to balance on multi-core.
Conclusions

Network programming in Haskell is fun thanks to user threads!

But GHC 6.12 is weak in network programming due to select.

GHC 6.14 would solve this problem. Let's enjoy user-thread network programming.

Prefork library could be used to balance processes on multi-core.
Links

- Mighttpd
  - http://www.mew.org/~kazu/proj/mighttpd/

- My github
  - http://github.com/kazu-yamamoto

- JL Smiley
  - http://jamlog.podzone.org/