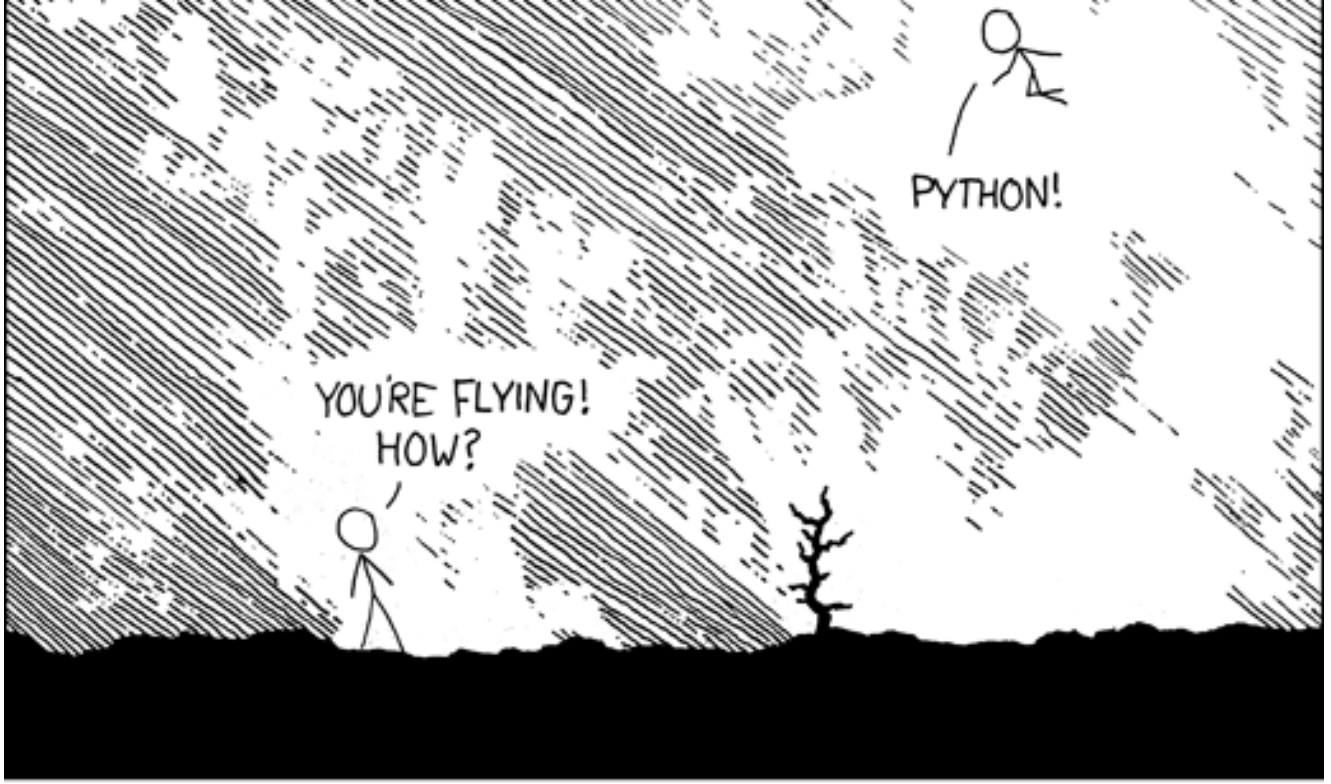
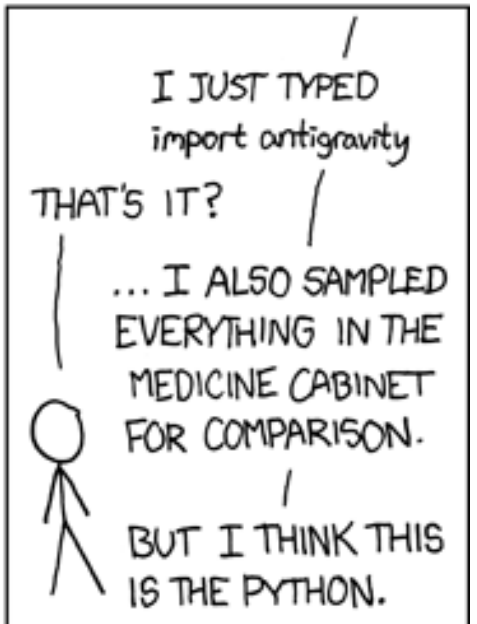
SENG 462 Tutorial #7

Chris Pearson
pearson@csc.uvic.ca

University of Victoria





Source:
xkcd.com

Testing

- EACH is worth 5% of your mark:
  - Correctness
  - Performance
  - Fault tolerance
  - Security

- Record EVERYTHING for your final project documentation
- Make graphs! Make tables!
  - Don't overload your reader. Impress your reader.

General Testing

- Dr. Neville will definitely ask about any graph that shows only one test run.
  - So… Do many!

- Results will vary every time
  - CPU load
  - Caching
  - Triggers

Correctness

- Make sure your results are repeatable
  - Write a quote server stub that always returns the same values
  - Run the system with known subsets of work data files
  - Make sure final result matches hand calculations
- Does the quote server always return the expected?
  - Do you always get the same username back?
  - Do you always get the correct stock label back?

Performance

- Discover where to start making performance improvements.
  - The worst bottlenecks
- Find where performance improvements are not directly effective
  - Code that can be improved 50%, but is only used 1% of the time
  - Places where improvements will simply cause other bottlenecks

Apache Example

- `man ab`

NAME

ab - Apache HTTP server benchmarking tool

SUMMARY

ab is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server. It is designed to give you an impression of how your current Apache installation performs. This especially shows you how many requests per second your Apache installation is capable of serving.

OPTIONS

-c concurrency

Number of multiple requests to perform at a time. Default is one request at a time.

-n requests

Number of requests to perform for the benchmarking session. The default is to just perform a single request which usually leads to non-representative benchmarking results.

Apache Example

- .../bin/ab -n 5000 -c 5 https://localhost:44443/

```
Server Software:        Apache/2.2.14
Server Hostname:        e104.seng.engr.uvic.ca
Server Port:            44443
SSL/TLS Protocol:       TLSv1/SSLv3,DHE-RSA-AES256-SHA,1024,256

Concurrency Level:      5
Time taken for tests:   164.140 seconds
Complete requests:      5000
Failed requests:        0
Write errors:           0
Total transferred:      1620000 bytes
HTML transferred:       220000 bytes
Requests per second:    30.46 [#/sec] (mean)
Time per request:       164.140 [ms] (mean)
Time per request:       32.828 [ms] (mean, across al
Transfer rate:          9.64 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       54  133  31.9    124     329
Processing:     1   31  31.2     18     207
Waiting:        0   26  24.4     18     191
Total:         60  164  51.6    144     461
```

```
Percentage of the requests
served within a certain time
(ms)
  50%    144
  66%    163
  75%    202
  80%    217
  90%    241
  95%    259
  98%    285
  99%    320
 100%    461 (longest request)
```

ab Ideas

- Concurrency level
  - Performance when less than number of Apache threads?
  - Performance when greater than number of Apache threads?
- Tweak number of Apache threads
  - Edit the httpd-mpm.conf
  - Where is the sweet spot for your application?
  - Too few = underperformance
  - Too many = underperformance

Example

- Apache with 2 threads:

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0  316 543.5      0    4766
Processing:    93  177  45.5    175     783
Waiting:       89  175  45.1    172     775
Total:         93  493 548.4    199    5041
```

- Apache with 10 threads:

```
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   31  49.3      0     210
Processing:   219  342  36.8    336     507
Waiting:      218  335  36.6    330     507
Total:        223  373  62.0    357     620
```

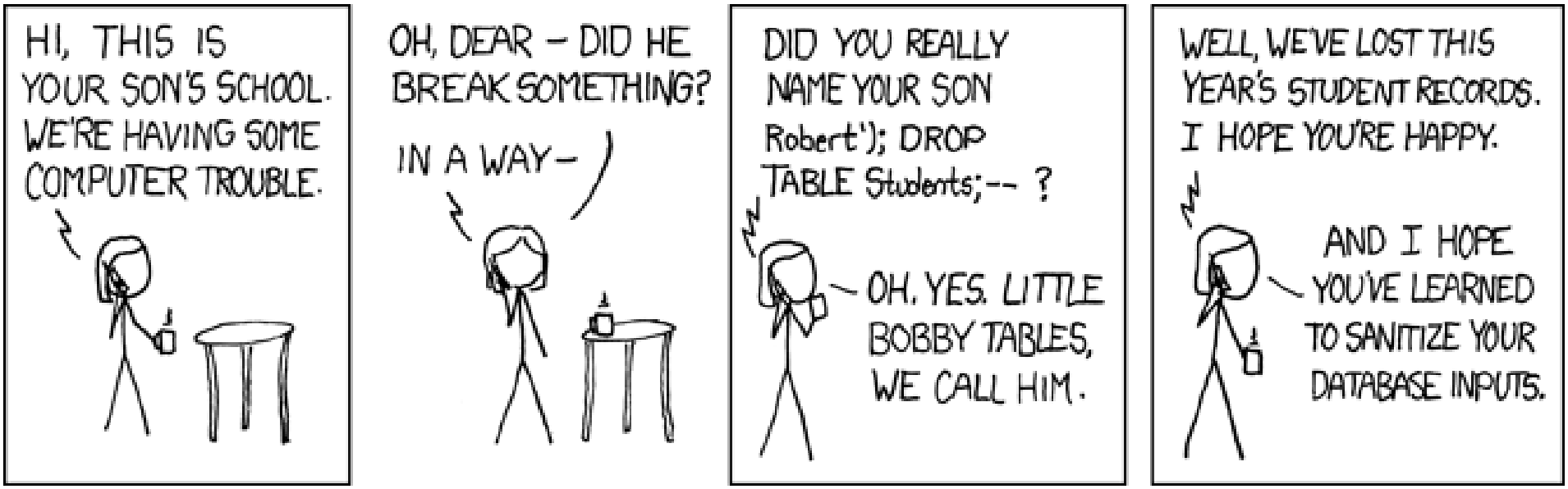
Fault Tolerance

- What happens if bad data is received?
  - errorEvent in your log
  - Will your servers handle it properly?
  - Test it! Record what happens
- What happens if a server fails?
  - Randomly disconnect servers
  - Record what happens
- Are the servers running properly?
  - Use a cron job set for every few minutes to check
  - This is a good argument for minimized downtime claims

Security

- Remember what Dr. Neville researches...
- Can regular users access admin functions?
  - Check it yourself
  - Set up software to check the logs
  - Is there a performance hit?
- What have you done to prevent malicious users?
  - What happens if random/malicious data is submitted to the web server?
  - SQL injections!

SQL Injections!



Source: xkcd.com

SENG 462 Tutorial #7

Chris Pearson
pearson@csc.uvic.ca

University of Victoria