

# PHP PERFORMANCE

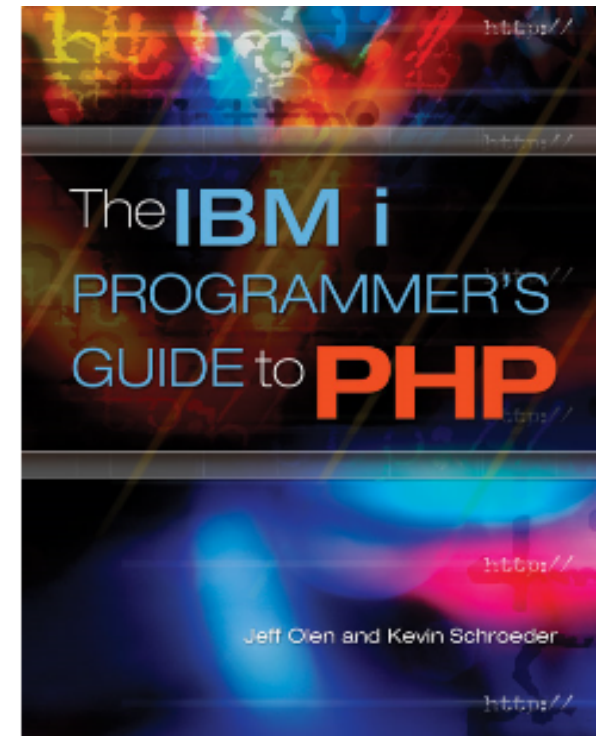
Principles and Tools

By Kevin Schroeder  
Technical Consultant  
Zend Technologies



# About me

- **Kevin Schroeder**
  - Consultant for Zend
  - Programmer
  - Sys Admin
  - Author
  - Etc.



# Things we'll look at

- **Database**
- **Network Latency**
- **IO Contention**
- **CPU utilization**
- **Network Connectivity**
- **Best Practices/Pit Falls/Things to consider**

# Performance – Database

- **Tools**
  - Zend Server – A distinct group of monitoring options for DB's
  - Inter-application query monitoring, usually via a DB adapter abstraction layer
- **Symptoms**
  - Pages take a long time to load with Apache/IIS/PHP using proportionally less of the CPU time
- **Causes**
  - The list could go on for  $1 \times 10^5$  slides

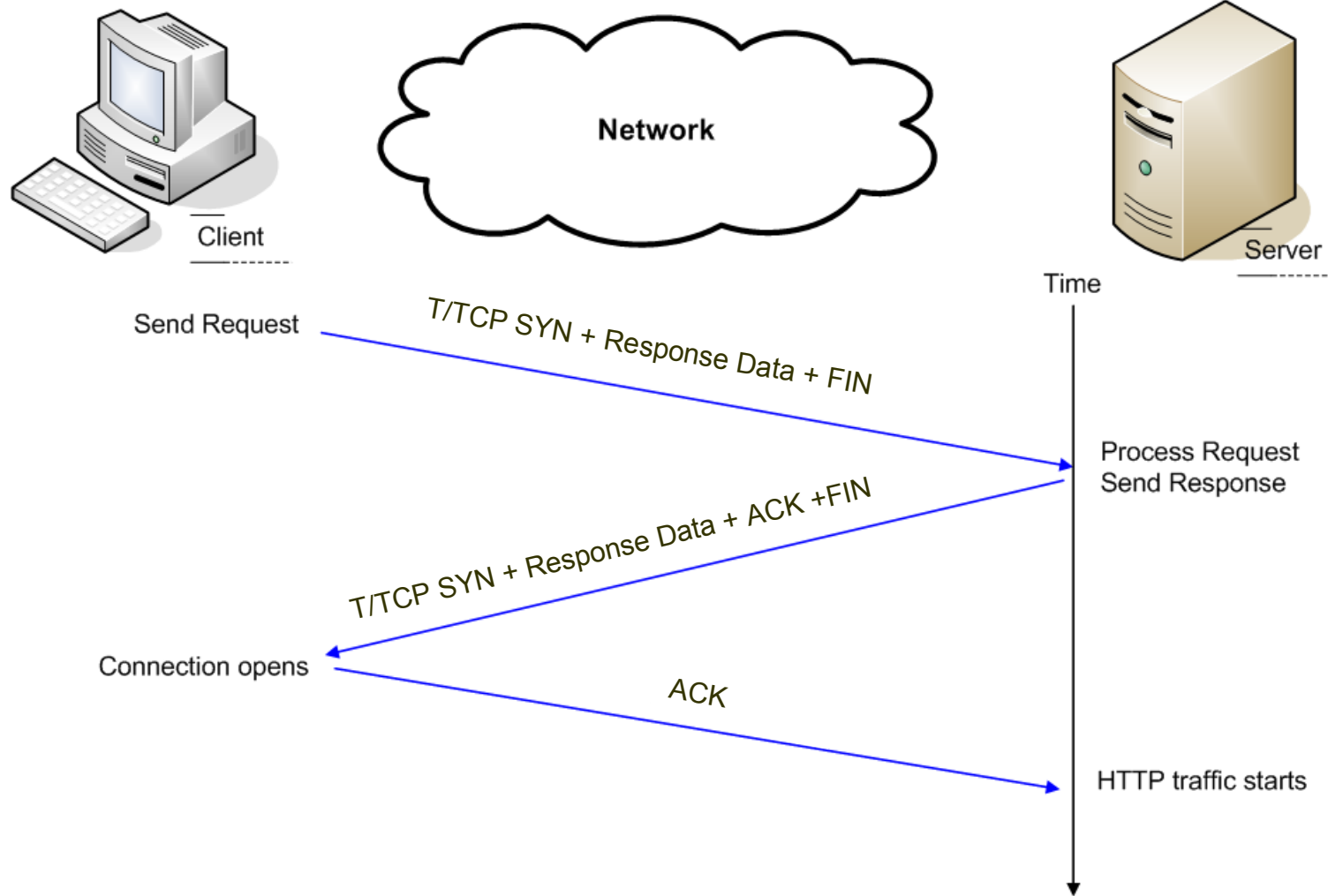
# Performance - Database

- **Possible remedies**
  - Simplify your queries
  - Make your queries more complex
  - Tune your database

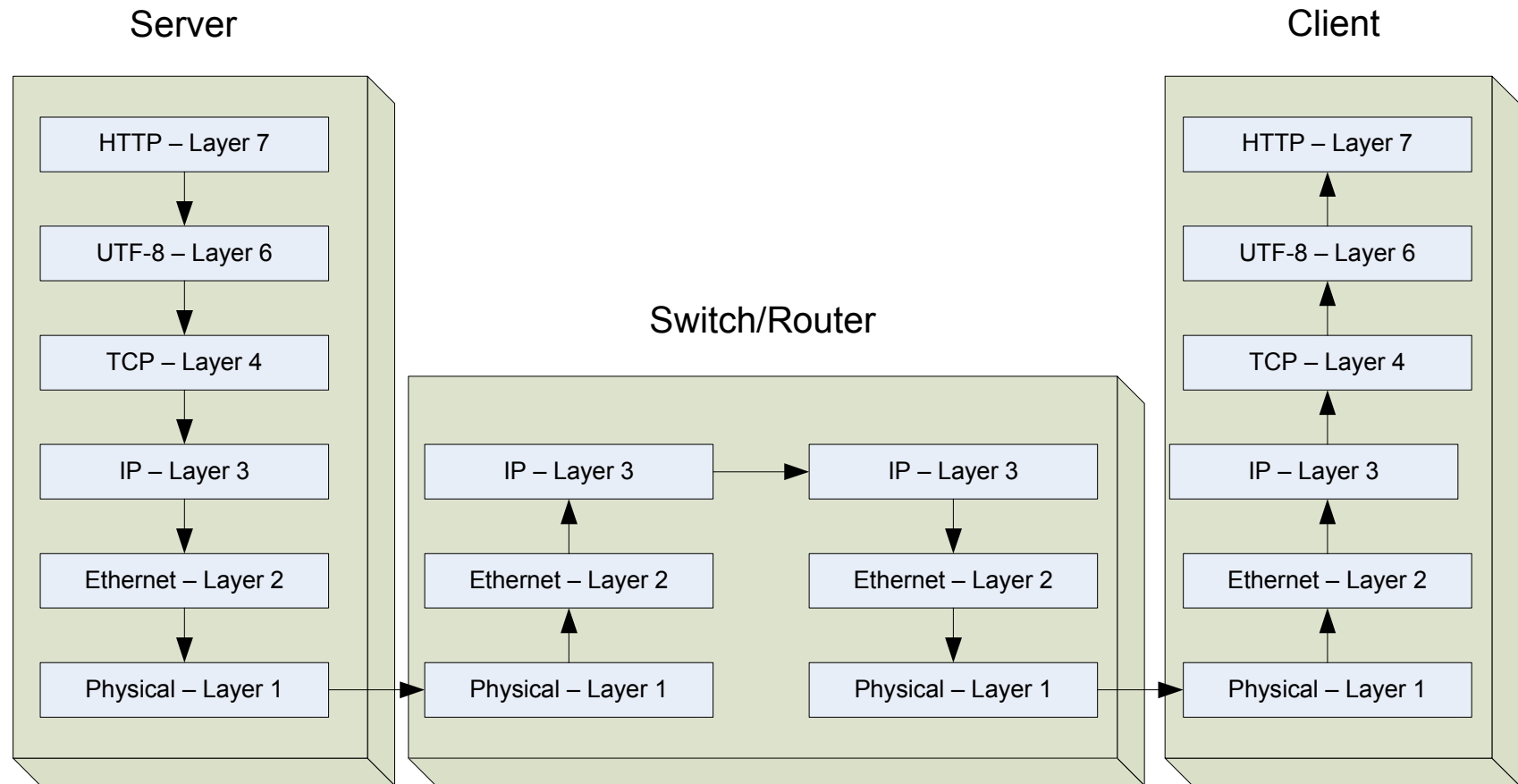
# Performance – Network Latency

- **Tools**
  - Zend Server – monitor slow socket function execution
  - Wireshark
  - Tcpdump
- **Symptoms**
  - Depends on where the latency is occurring
- **Causes**
  - Network Saturation (probably not too likely)
  - DNS Reverse Lookups
  - TCP Handshakes
  - High number of “hops”
  - ... and much more

# The TCP handshake



# Hops (not the good kind)



# Performance – IO Contention

- **Tools**
  - Zend Server (in extreme cases, long `fopen()` calls)
  - `vmstat` & `top` (Linux)
  - System Internals Process Explorer (Windows)
- **Symptoms**
  - Unpredictable and difficult to reproduce page load times
- **Causes**
  - Too few spindles serving too much data
  - High write concurrency
  - Insufficient free system RAM

# Performance – CPU Utilization

- **Tools**
  - top (Linux)
  - System Internals Process Explorer (Windows)
- **Symptoms**
  - Page load times either correspond directly to CPU utilization or are consistently slow
- **Causes**
  - Bad code or over-architected solutions
  - Excessive recursion
  - Excessive IO
  - High server utilization
    - too few servers serving too much traffic

# Performance – CPU Utilization

- **There are often 2 different paths this can follow**
  - User time
    - Time spent executing the code that you wrote
  - System time
    - Time spent running functionality that your code calls, such as networking, file system, etc.

# Performance – Network Connectivity

- **Tools**

- Zend Server – monitor failed calls to `socket_connect()`, `fsockopen()` or `fread()`
- Wireshark
- Tcpdump

- **Symptoms**

- Database or web service calls take long to execute
- If read functions die unexpectedly, bad network hardware might be a cause

# Choosing the right architecture

- If you deal with large collections of data, be careful of various ORM techniques.

This also pertains to the cloud

	Limit 100	Limit 500	Limit 1000	Unlimited (250k)
Active Record	0.056	0.27	0.55	13.3
Collection	0.002	0.008	0.01	0.2
Difference (Col * Dif) = AR	28	33.75	55	66.5

- **Beware of magic!**
  - `__get`, `__set`, `__call`
  - Situations where they are often used
    - When representing another object that it has no knowledge of - I.e. Soap requests, COM objects, Java objects, database tables
    - When there are property-level permissions on an object's properties. Keeping the properties private and using `__get/set` to determine access
  - If used sparingly they won't affect you, but many modern applications have lots of recursion or large underlying architectures that can have unpredicted consequences

# !Preg

- **Use preg\_match for data checking only when you have to**
  - `stripos('http://', $website)` is over twice as fast as `preg_match('/http:\/\//i', $website)`
  - `ctype_alnum()` is almost 5 times as fast as `preg_match('/^\s*$/')`;
  - Casting `“if ($test == (int)$test)”` is well over 5 times faster than `preg_match('/^\d*$/')`

# Code Acceleration

- **From a test on Wordpress**
  - Without acceleration
    - Requests per second: 2.28
    - Time per request: 437.766 ms
  - With acceleration
    - Requests per second: 8.11
    - Time per request: 123.377 ms
  - 4-fold increase in performance by flipping a (free) switch
  - Zend Server CE is free and has this. You have no excuse!!

# Do you queue?

- **Many applications process long running requests inline**
- **PHP can only serve one request on one process at one time**
- **Offloading long running processes to a queuing system can help make your front end servers scale more vertically**

# Non-PHP PHP performance rules

- **PHP performance is not the only factor in determining the overall performance of a website. Here are some suggestions**
  - Make fewer HTTP requests
  - Use an Expires header
  - Gzip cached content
  - Put stylesheets at the top
  - Put Javascript at the bottom
  - Reduce DNS lookups
  - Avoid redirects

# Non-PHP PHP performance rules

- **Be aware of your page size**
  - Microsoft – 238kb – 4.4 seconds
  - Oracle – 104kb – 861ms
  - IBM – 640 kb – 4.6 seconds
  - Zend – 183kb – 2.4 seconds
- **There is a fairly direct relationship between the amount of data on your web page and the render time**

# Non-PHP PHP performance rules

- **Similarly, be careful when dealing with lots of Ajax data**
- **The browser needs to process that data**
- **With large data sets the browser may take a while to update the DOM**
- **2 options**
  - Use staggered loading
  - Pass rendered content to the browser and attach with innerHTML

# Fin

**Kevin Schroeder**  
**kevin@zend.com**



The PHP Company