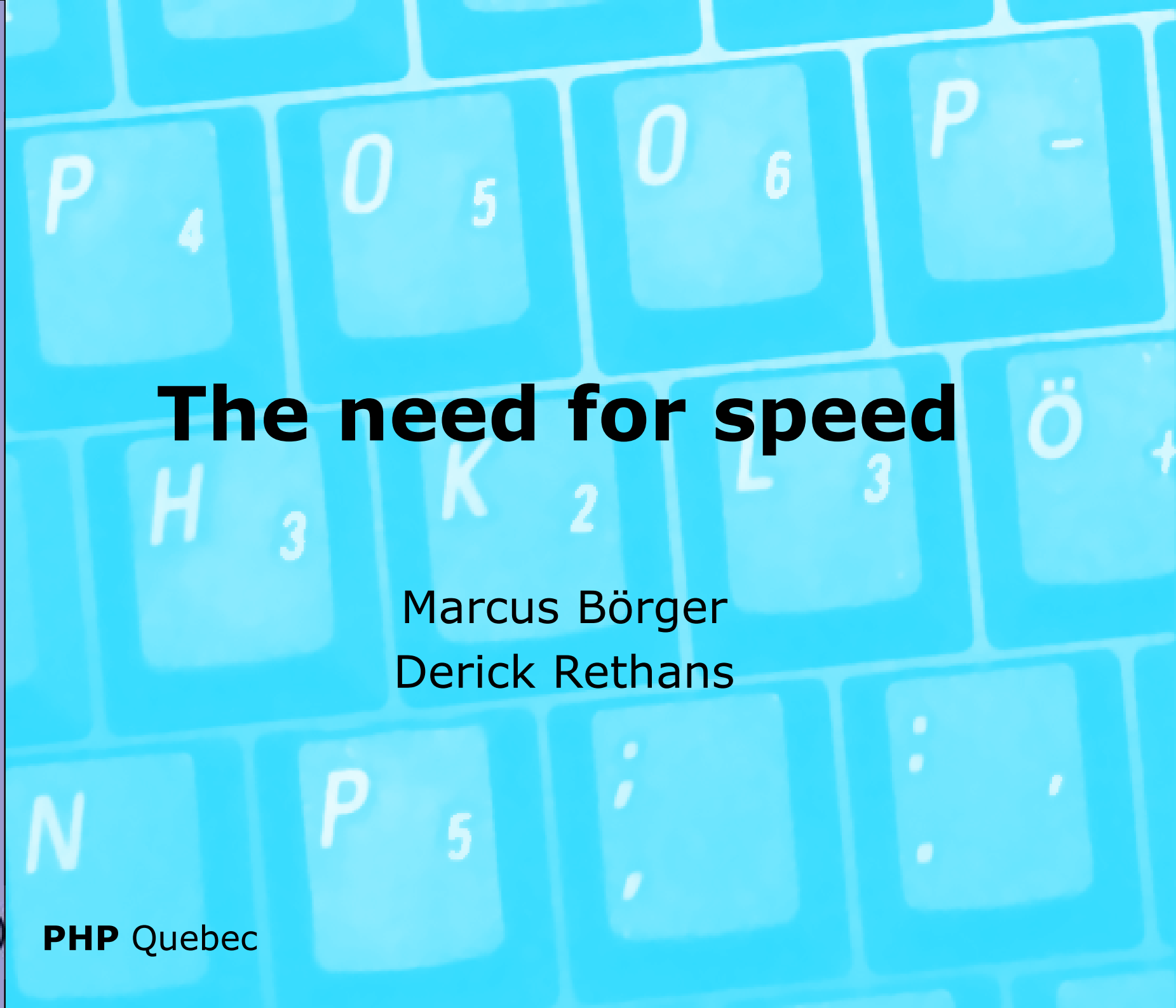


The need for speed

Marcus Börger
Derick Rethans

PHP Quebec

The need for speed

- ☑ General aspects
  - Communication
  - Hardware
  - Operating system
- ☑ How to use PHP
  - As a web scripting language
  - As a template system
  - As a RAD tool
  - The Rasmus way
- ☑ What to do and what not to do with PHP

General aspects

☑ Do not loose your focus

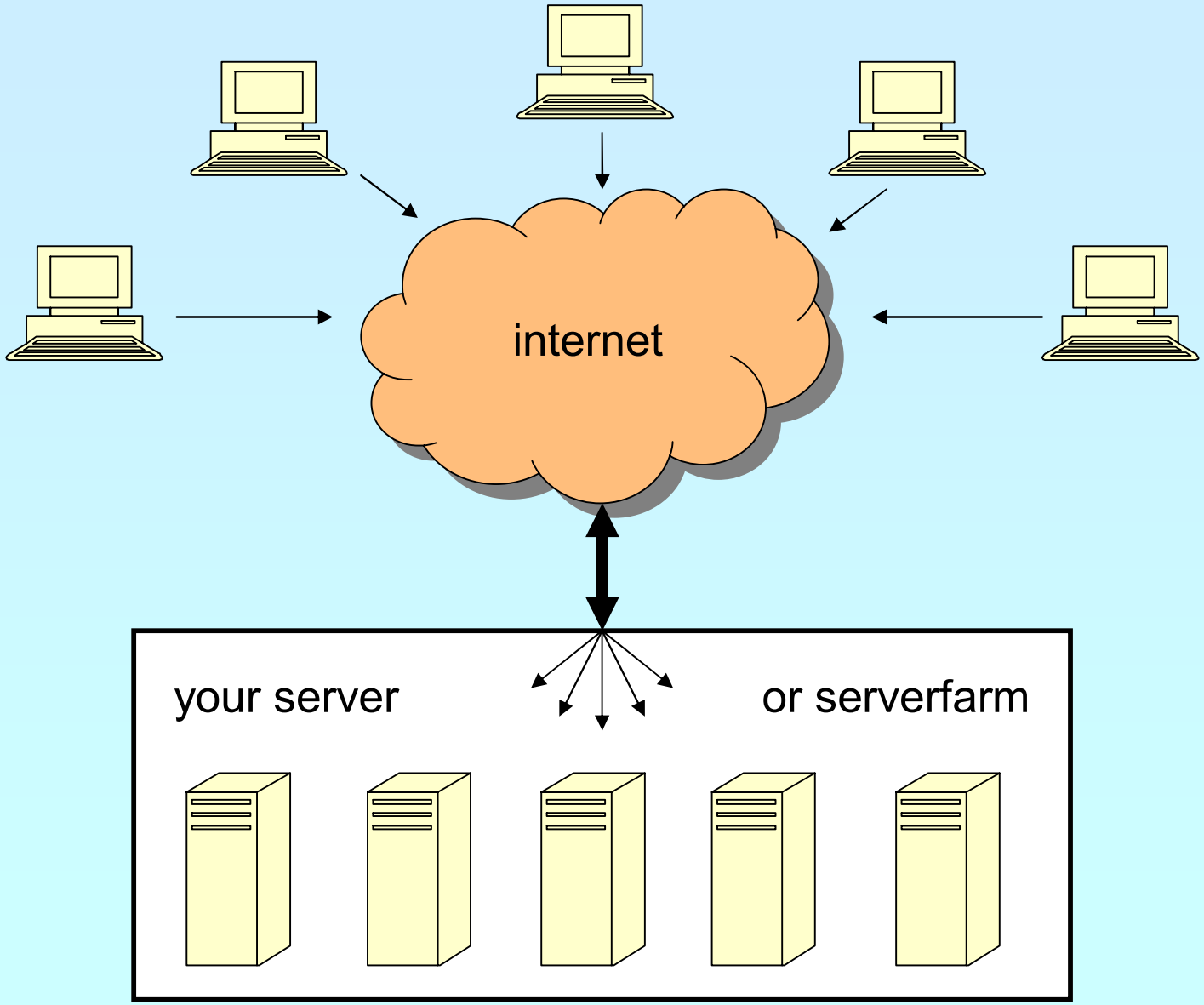
- Think before you do anything
- Always check you are still on track
- Estimate the time and money you (still) have
- Estimate the time and money you (still) need

- Are you using the right tools?
  - Is PHP the correct choice?
  - After all is a web application the right thing?

- Are you using the right algorithms?
  - Is there a better way?

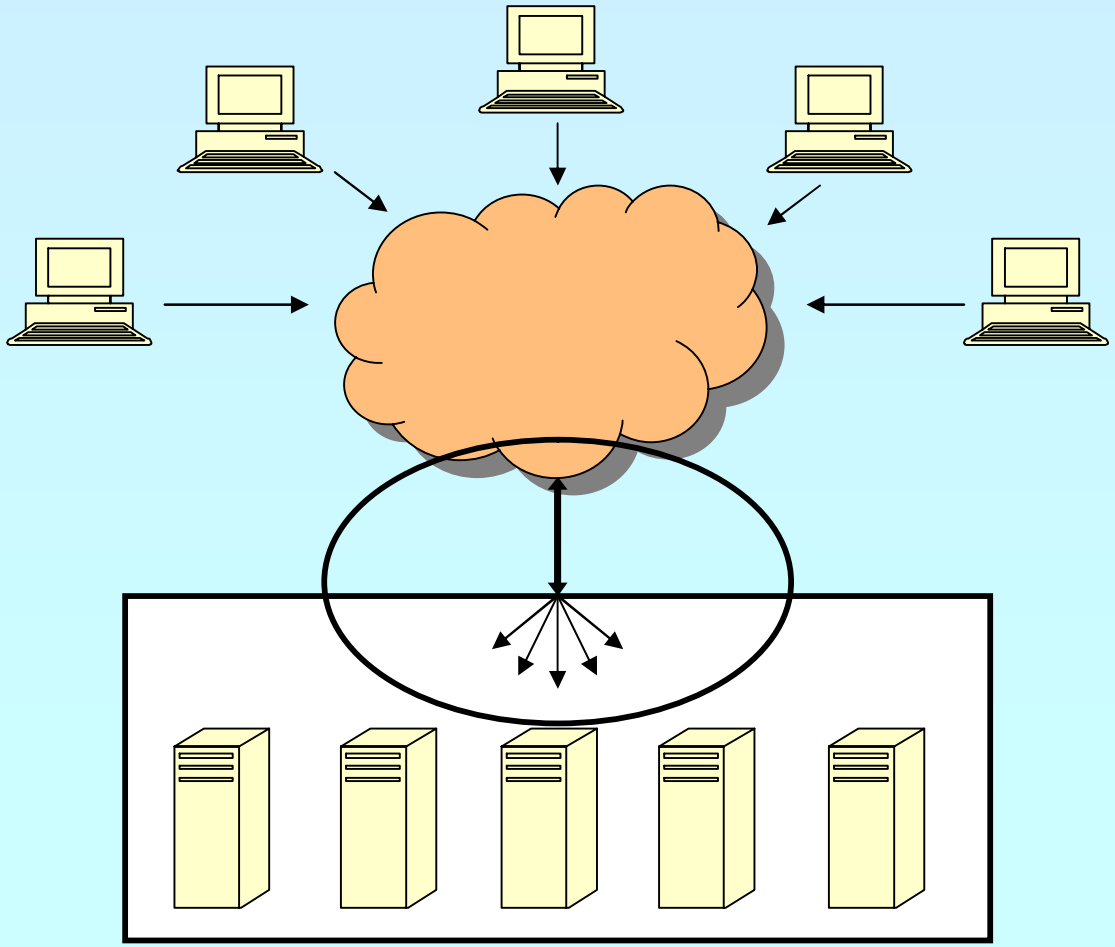
- Know your environment
- Know your team

Communication



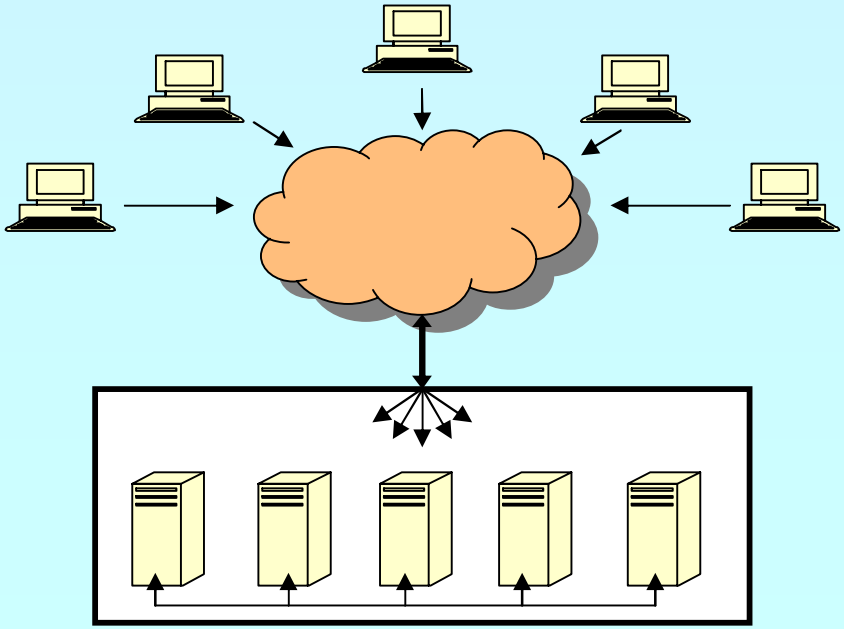
Communication

- The sum is smaller than the whole
  - No need to apply more servers if no more bandwidth is available



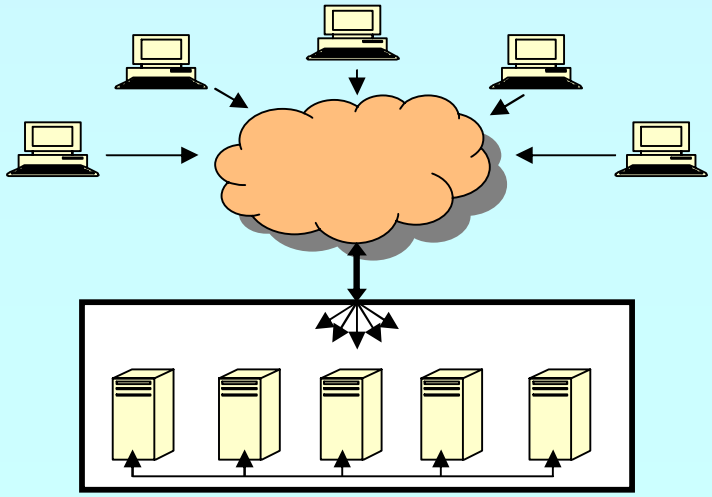
Communication

- ☑ The sum is smaller than the whole
  - ➢ No need to apply more servers if no more bandwith is available
- ☑ A prepared DDoS can put down anything
- ☑ Applying more servers means they communicate



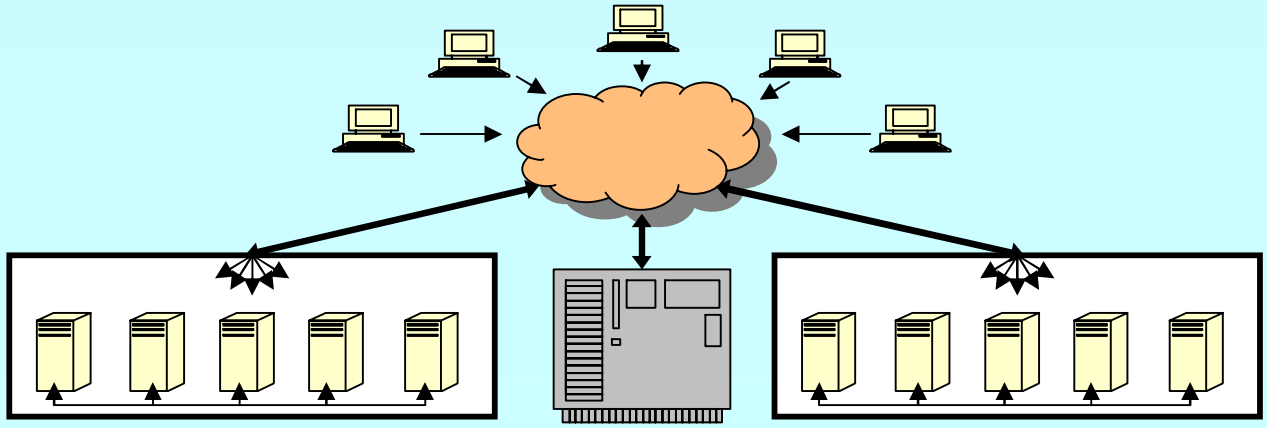
Communication

- ☑ The sum is smaller than the whole
  - No need to apply more servers if no more bandwith is available
- ☑ A prepared DDoS can put down anything
- ☑ Applying more servers might help
  - They will communicate
  - You need more software
  - You have more points of failure



Communication

- ☑ The sum is smaller than the whole
  - ➢ No need to apply more servers if no more bandwith is available
- ☑ A prepared DDoS can put down anything
- ☑ Applying more servers might help
  - ➢ They will communicate
  - ➢ You need more software
  - ➢ You have more points of failure
- ☑ New ideas can help



Hardware

- ☑ Every single hardware piece is a point of failure
  - ☑ Avoid single point of failures
  - ☑ Use the hardware as specified (speed, temperature)
  - ☑ Don’t use it to emulate other hardware
  - ☑ Don't use it to imitate other hardware

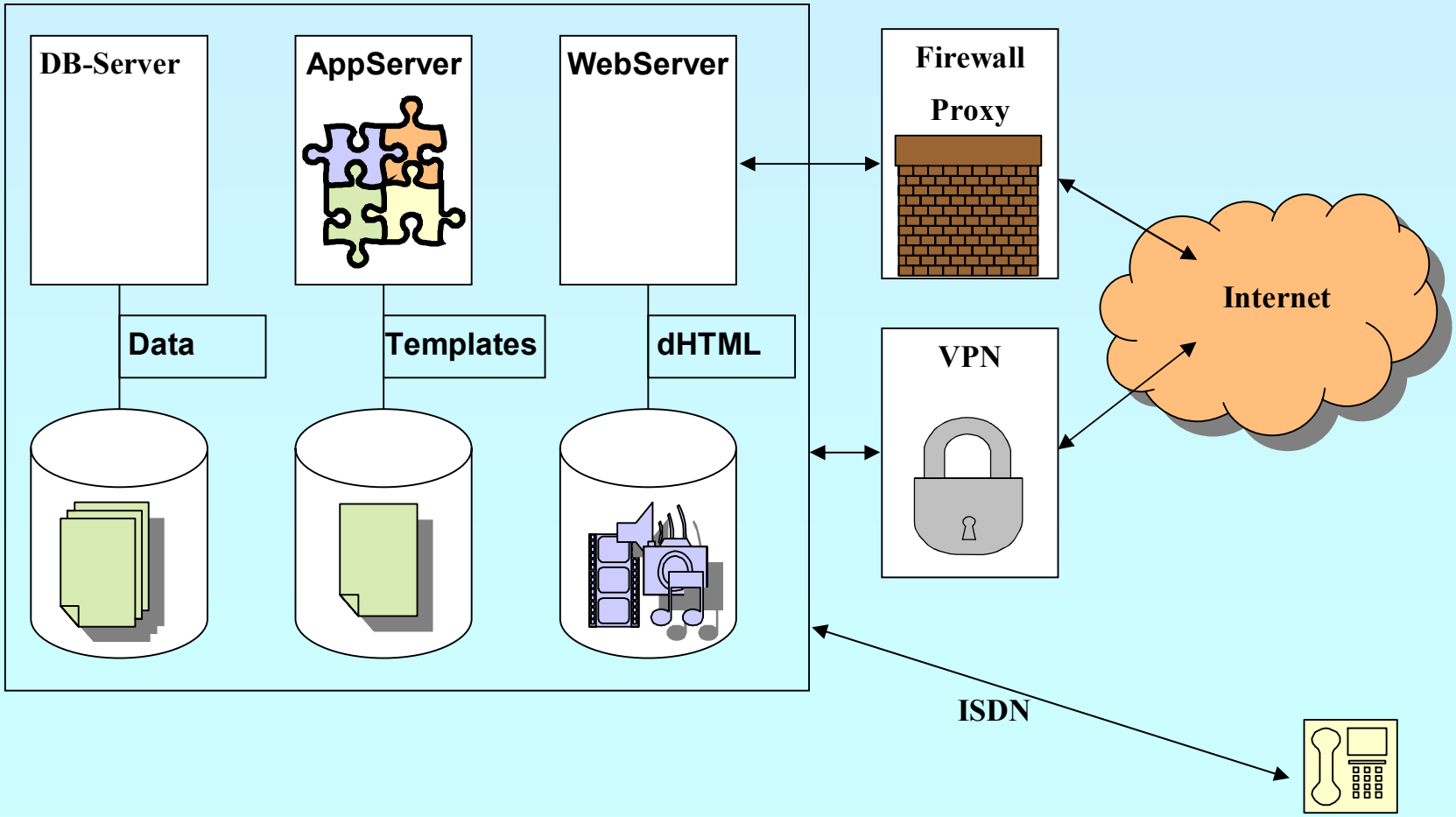
  - ☑ If you don't have enough knowledge give it away

Operating system

- ☑ Choose the OS based on
  - ☑ your hardware
  - ☑ your software
  - ☑ what you are going to do

Architecture

- ☑ Apply specialization



Database Server

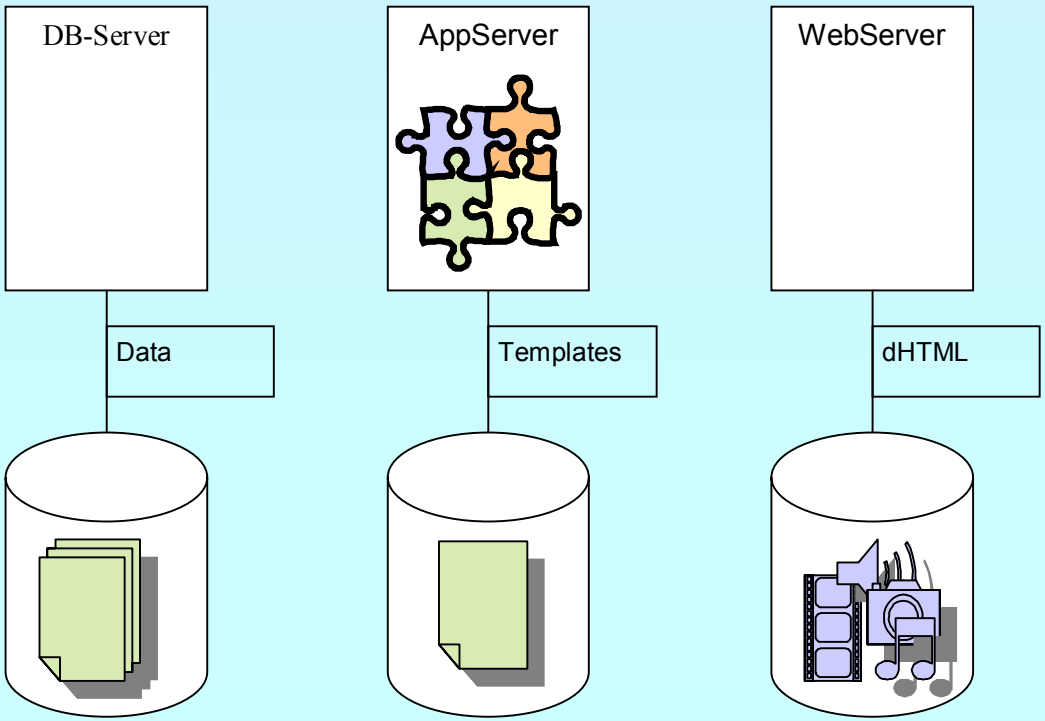
- ☑ What kind of data
- ☑ What size does your data have
- ☑ Who is responsible for data integrity
- ☑ Who is responsible for security
- ☑ Does the database need its own logic

Application Server

You want dependency injection?

You need inversion of control?

☞ PHP would need state first



Web server

- ☑ Apache
  - ☑ Suitable for nearly all needs
- ☑ Microsoft IIS
  - ☑ Perfect when the rest is also Microsoft
  - ☑ Threadsafty issues
  - ☑ Not the major/focused development platform
- ☑ Zeus
  - ☑ Very fast

Web server

- ☑ TUX - kernel-based web server
  - ☑ Virtual Host support.
- ☑ thttpd - tiny/turbo/throttling HTTP server
  - ☑ Non-blocking I/O is good.
  - ☑ Throttling capabilities.
- ☑ lighttpd
  - ☑ On the fly compression.
  - ☑ Excellent virtual host support.

Web server

Plenty of CPU power but limited bandwidth

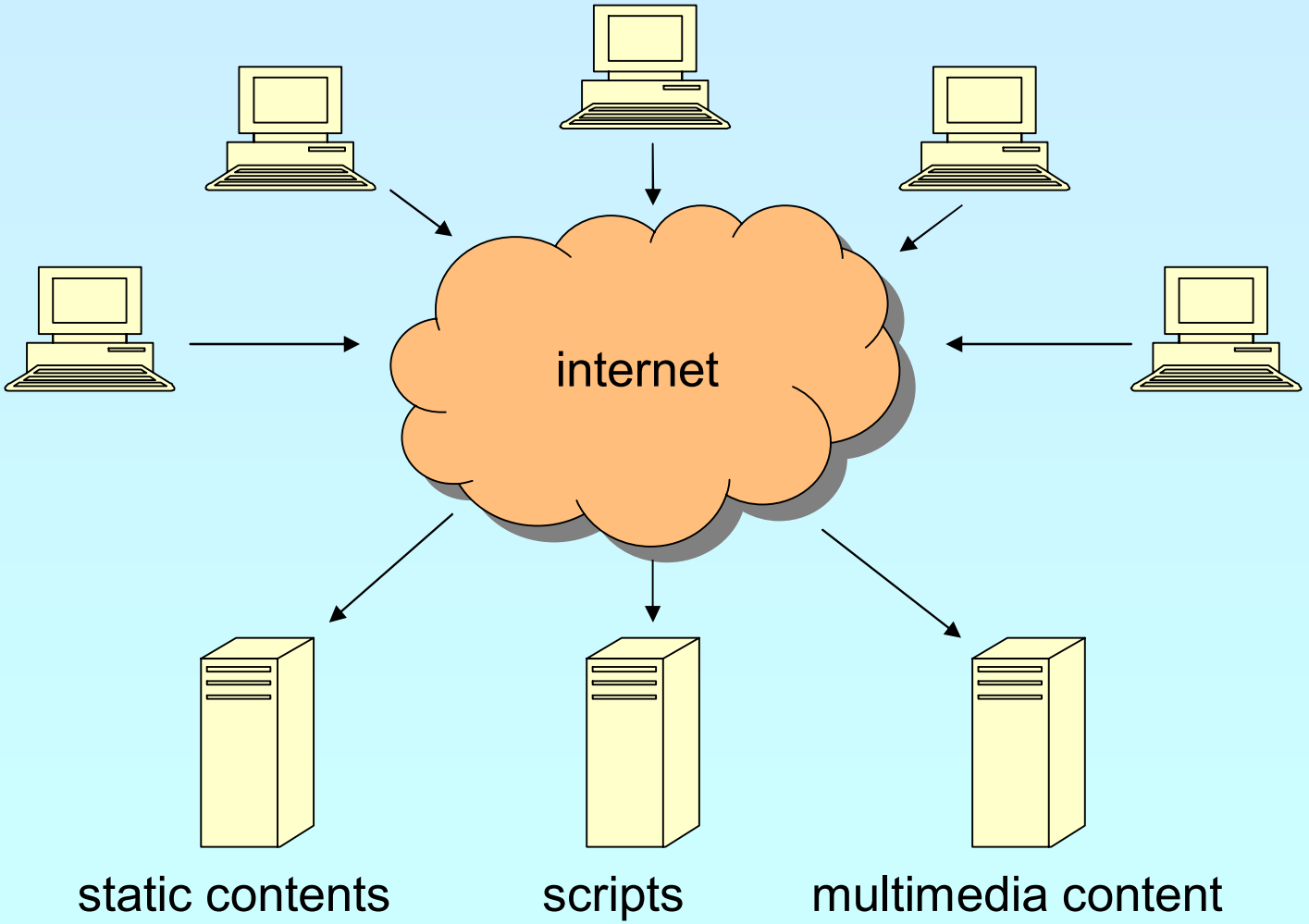
- ☞ Turn on output compression

Much bandwidth but limited CPU power

- ☞ Do not use output compression

Web Server

- ☑ Use different web servers for different things



What is PHP

PHP is a scripting language specifically designed to help developers solve web problems, it works by embedding sections of code within HTML blocks.

PHP Advantages

- Easy to learn
- Targeted, built-in functions for web developers
- Good introduction to programming
- Configurable
- Simple extension API
- PEAR
- Runs britneyspears.com

PHP Disadvantages

- Focused on the Web environment
- Poor OO support until PHP 5
- Configurability Hurts Portability
- Easy for beginning users,
- Easy for beginning users to make mistakes

PHP - As web scripting language

- ☑ Every page is its own PHP script
  - Flexible and easy
    - Independent scripts by independent programmers
  - Hard to apply general tasks to all pages
    - Includes can help
    - CSS can help

PHP - As a template system

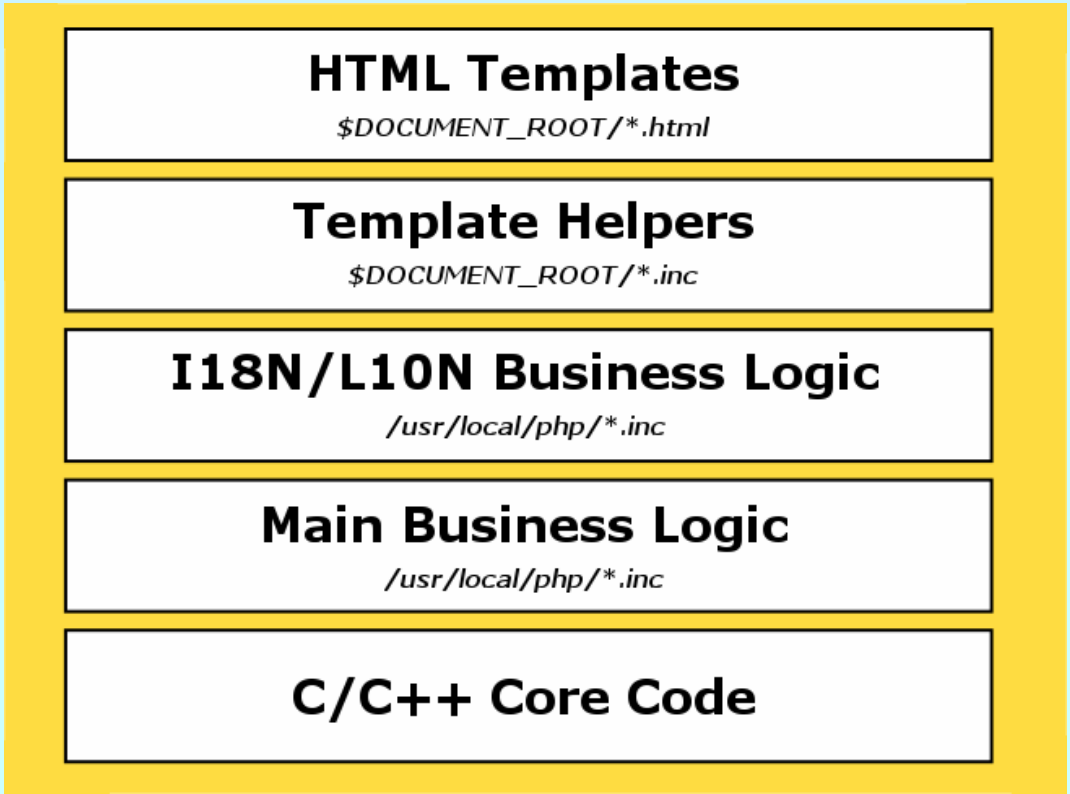
- ☑ PHP was developed as a template system
  - ☑ PHP can be used as template system
  - ☑ PHP can be the language to develop a template system

PHP - As a RAD tool

- ☑ No PHP in your real applications
  - ☑ Test with PHP
  - ☑ Implement in another language

PHP - The Rasmus way

- ☑ Small basic PHP scripts
- ☑ Small include files to solve general aspects
- ☑ Include files for the business logic
- ☑ Specialized extensions for the actual work



Break

References

- ☑ Copying a variable takes time

- ☞ Learn when PHP needs to copy
- ☞ Learn about references

References

A famous PHP 4 rule:

If your code doesn't work spread some '&'s into it

If it still doesn't work use more '&'

- ☞ Understand references

References

- References are aliases
  - If you change one you change all others

```
<?php           // empty global table
$a = 25;        // creates a zval
$b = $a;        // creates a pointer to $a
$b = 42;        // makes $b a copy of $a and changes it
$c = $a;        // create another pointer to $a
$d = &$a;       // split/copy $a, creates $d as a reference to $a
$c = 43;        // change $c only
$d = 0;         // changes $d and hence $a
?>
```

References

- ☑ Variables are normally copied on function calls

```
<?php

function test($a)
{
}

$a = array(25);     // creates a global zval


test($a);           // creates a new symbol table, copies $a

?>
```

References

- ☑ Variables can be passed as references

```
<?php

function test(&$b)
{
   $b[] = 42;     // adds a new value to local $b = global $a
}

$a = array(25);  // creates a global zval


test($a);        // creates a new symbol table

?>
```

References

- ☑ Variables are normally copied on return

```
<?php

function test(&$b)
{
   return $b;
}

$a = array(25);

$b = test($a);      // $b is a new value, copied on return

?>
```

References

- ☑ Functions can return aliases

```
<?php

function &test(&$b)
{
   return $b;
}

$a = array(25);

$b = test($a);      // $b is a new value, copied after return

?>
```

References

- ☑ Functions can return aliases
- ☑ Explicit use of the returned reference is needed

```
<?php

function &test(&$b)
{
   return $b;
}

$a = array(25);

$b = &test($a);    // $b is a reference to $a

?>
```

References

- Objects should always be references
  - In PHP 5 they are object-references

```
<?php
class test
{
    function factory() {
        return new test();
    }
}

$obj = test::factory();
?>
```

References

- Objects should always be references
  - In PHP 5 they are object-references
  - In PHP 3 and 4 you have to take care yourself

```
<?php
class test
{
    function &factory() {
        $a = &new test();
        return $a;
    }
}

$obj = &test::factory();
?>
```

References

- Most internal functions don't use references
  - This is to allows you to pass arrays and strings without copying them into a variable first

```
<?php
$a = array_fill(0, $cnt, 'foo');
array_key_exists($i, $a);    // is_ref == 0, refcount == 1
$b = $a;
array_key_exists($i, $a);    // is_ref == 0, refcount == 2
array_key_exists($i, &$a);  // is_ref == 0, refcount == 2
unset($b);
$b =& $a;                    // making a reference, but not using it
array_key_exists($i, $b);    // is_ref == 1, refcount > 1 (pass as var)
array_key_exists($i, &$a);  // is_ref == 1, refcount > 1 (pass as ref)
unset($b);
array_key_exists($i, $a);    // is_ref == ?, refcount == 1
?>
```

Use the right tool
For the right problem

- ☑ Use OOP where appropriate not where nice
- ☑ Use layers not because it is easy or looks nice
- ☑ Use abstraction if derived or used often
- ☑ Use indirection if it is of any advantage

The 80 / 20 rule

- ☑ 80% of your code takes less than 20% runtime
- ☑ You don't need to optimize anything in the 80%
- ☑ Find out which are the 20% to optimize

Stop Optimizing

- ☑ Don’t get overexcited about optimization
- ☑ Sometimes it is cheaper and more efficient
  - ☑ to buy another server
  - ☑ to increase bandwidth
  - ☑ To buy faster software

Conférence PHP Québec

THANK YOU

http://somabo.de/talks/

http://derickrethans.nl/talks.php