CakePHP Training

Mastering advanced topics

Course Outline

- Major Topics
  - Authorization
  - Advanced Pagination
  - Routing
  - Logging
  - Internationalization
- Deploy on the cloud
- Master - Slave setups

Topic:

Authorization

- Built-in ACL (very specific use cases)
- Custom controller logic (isAuthorized)
- Custom Authorization object

Custom Authorization

- Objects living in Controller/Component/Auth
- Implement the authorize() method
- Return whether the user has permissions or not

```php
public function authorize($user, CakeRequest $request) {
        if ($user['admin']) {
                return true;
        }
        $role = Configure::read('Permissions.' . $this->action($request));
        if (!$role) {
                return false;
        }
        if ($role === '*') {
                return true;
        }
        $models = explode(',', Inflector::camelize($role));
        foreach ($models as $model) {
                $exists = ClassRegistry::init($model)->find('count', array(
                        'conditions' => array(
                                'user_id' => $user['id'],
                                'course_id' => $request->params['pass'][0]
                        )
                ));
                if ($exists) {
                        return true;
                }
        }
        return false;
}
```

```php
<?php

App::uses('BaseAuthorize', 'Controller/Component/Auth');
class TrainingAuthorize extends BaseAuthorize {

/**
 * Checks user authorization.
 *
 * @param User $user Active user
 * @param CakeRequest $request
 * @return boolean
 */
    public function authorize($user, CakeRequest $request) {
        if ($user['role'] === User::ADMIN) {
            return true;
        }

        if ($request['controller'] === 'users' && $request['action'] !== 'delete') {
            return $user->getId() == $request->params['pass'][0];
        }

        if ($request['controller'] === 'class_rooms' && $request['action'] === 'view') {
            $course = ClassRoom::find($request->params['pass'][0]);
            return !$course || $course->isEnrolled($user);
        }

        if ($request['controller'] === 'class_rooms' && in_array($request['action'], array('mycourses', 'join'))) {
            return true;
        }

        return false;
    }

}
```

Advanced Pagination

and custom find methods

```php
class Photo extends AppModel {

    public $findMethods = array('recent' => true);

    protected function _findRecent($state, $query, $results = array()) {
        if ($state == 'before') {
            $query['conditions']['Photo.created >='] = date('Y-m-d', strtotime('-2 days'));
            return $query;
        }
        return $results
    }
}
```

$this->Photo->find('recent');

```php
class PhotosController extends AppController {

    public $components = array(
        'Paginator' => array(
            'settings' => array('recent')
        )
    );

/**
 * index method
 *
 * @return void
 */
    public function index() {
        $this->Photo->recursive = 0;
        $this->set('photos', $this->paginate());
    }
```

Paginating a custom find

```php
function paginateCount($conditions = array(), $recursive = 0, $extra = array()) {
    $parameters = compact('conditions');
    $find = '_findCount';
    if (isset($extra['type'])) {
        $extra['operation'] = 'count';
        $find = '_find' . Inflector::camelize($extra['type']);
        $params = $this->$find('before', array_merge($parameters, $extra));
        unset($params['fields']);
        unset($params['limit']);
        return $this->find('count', $params);
    }
    return $this->find('count', array_merge($parameters, $extra));
}
```

Nice trick or paginating complex custom finds

Custom routing

Is not as hard as it looks like

```
Router::connect('/photos/:year/:month/:day', array('controller' => 'photos', 'action' => 'archive'), array(
    'year' => Router::YEAR,
    'month' => Router::MONTH,
    'day' => Router::DAY,
    'pass' => array('year', 'month', 'day')
));
Router::connect('/photos/:year/:month', array('controller' => 'photos', 'action' => 'archive'), array(
    'year' => Router::YEAR,
    'month' => Router::MONTH,
    'pass' => array('year', 'month')
));
Router::connect('/photos/:year', array('controller' => 'photos', 'action' => 'archive'), array(
    'year' => Router::YEAR,
    'pass' => array('year')
));
```

Custom Route Classes

- Help you make any type of route you can think of
- Implement parse() and match()
- Use the class in the third param of Router::connect()
- Be wary of expensive calls or operations

```php
class SlugRoute extends CakeRoute {

    function parse($url) {
        $params = parent::parse($url);
        if (empty($params)) {
            return false;
        }
        App::import('Model', 'Post');
        $Post = new Post();
        $count = $Post->find('count', array(
            'conditions' => array('Post.slug LIKE ?' => $params['slug'] .'%'),
            'recursive' => -1
        ));
        if ($count) {
            return $params;
        }
        return false;
    }

}
```

Topic:

Logging

When to log?

- Always, but specially when dealing with async processes
  - Dealing with money
  - Processing emails
  - Background stuff
- CakeLog acts like event dispatcher
- How to log
  - CakeLog::write()
  - Use performant engines
  - Avoid file log
- All log engines are notified of logged message

Using CakeLog

```
CakeLog::write($level,  $message);
```

```php
App::uses('ClassRegistry', 'Utility');
App::uses('CakeLogInterface','Log');
class DatabaseLogger implements CakeLogInterface{

	/**
	* Model name placeholder
	*/
	var $model = null;

	/**
	* Model object placeholder
	*/
	var $Log = null;

	/**
	* Contruct the model class
	*/
	function __construct($options = array()){
		$this->model = isset($options['model']) ? $options['model'] : 'DatabaseLogger.Log';
		$this->Log = ClassRegistry::init($this->model);
	}

	/**
	* Write the log to database
	*/
	function write($type, $message){
		$this->Log->save(array(
			'type' => $type,
			'message' => $message
		));
	}
}
?>
```

Logging Emails

```php
class EmailConfig {

    public $default = array(
        'transport' => 'Mail',
        'from' => 'you@example.com',
        'email' => true
    );

}
```

Logged emails can be found in tmp/log/email.log

Topic:

I I8n

Built-in features

- Translation functions (__*())
- Extract shell
- Translate behavior
- Time localization

Up to you...

- Language in URL
- language switching

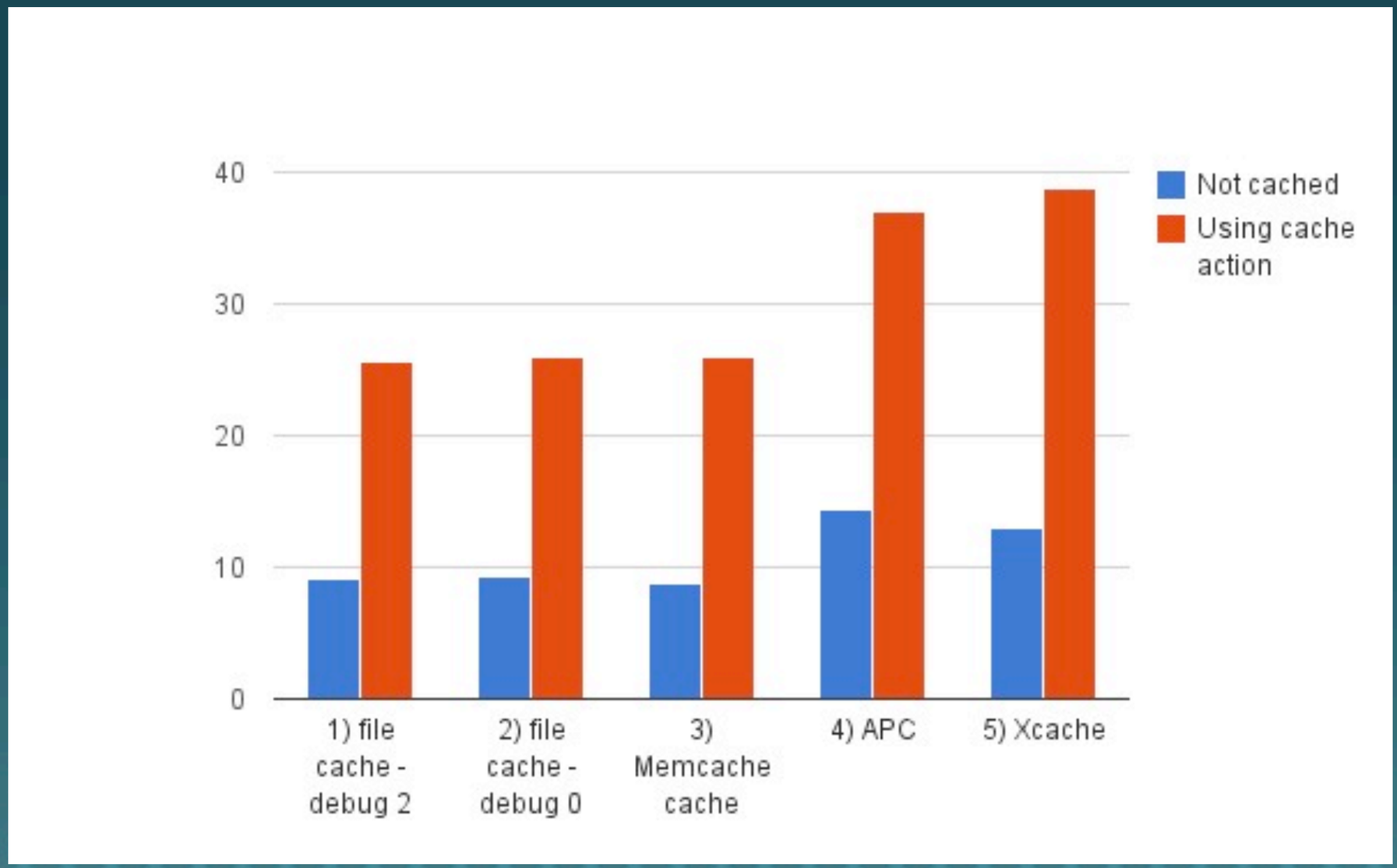
Topic:

The Cloud

Some considerations

- Do not rely on filesystem being there!!
- Use a shared cache
- Use shared sessions
- Use shared database
- Automate deploy and scaling

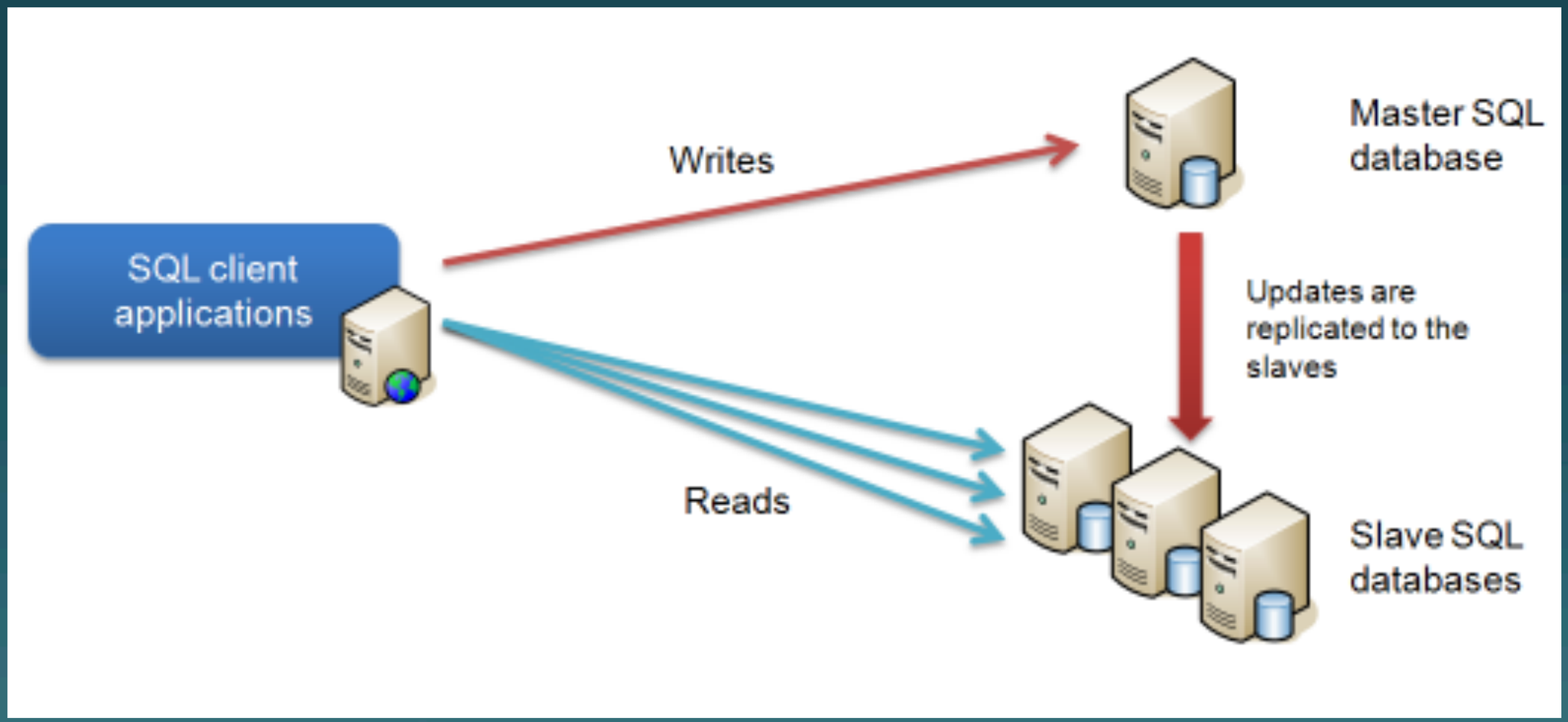
How to choose your cache



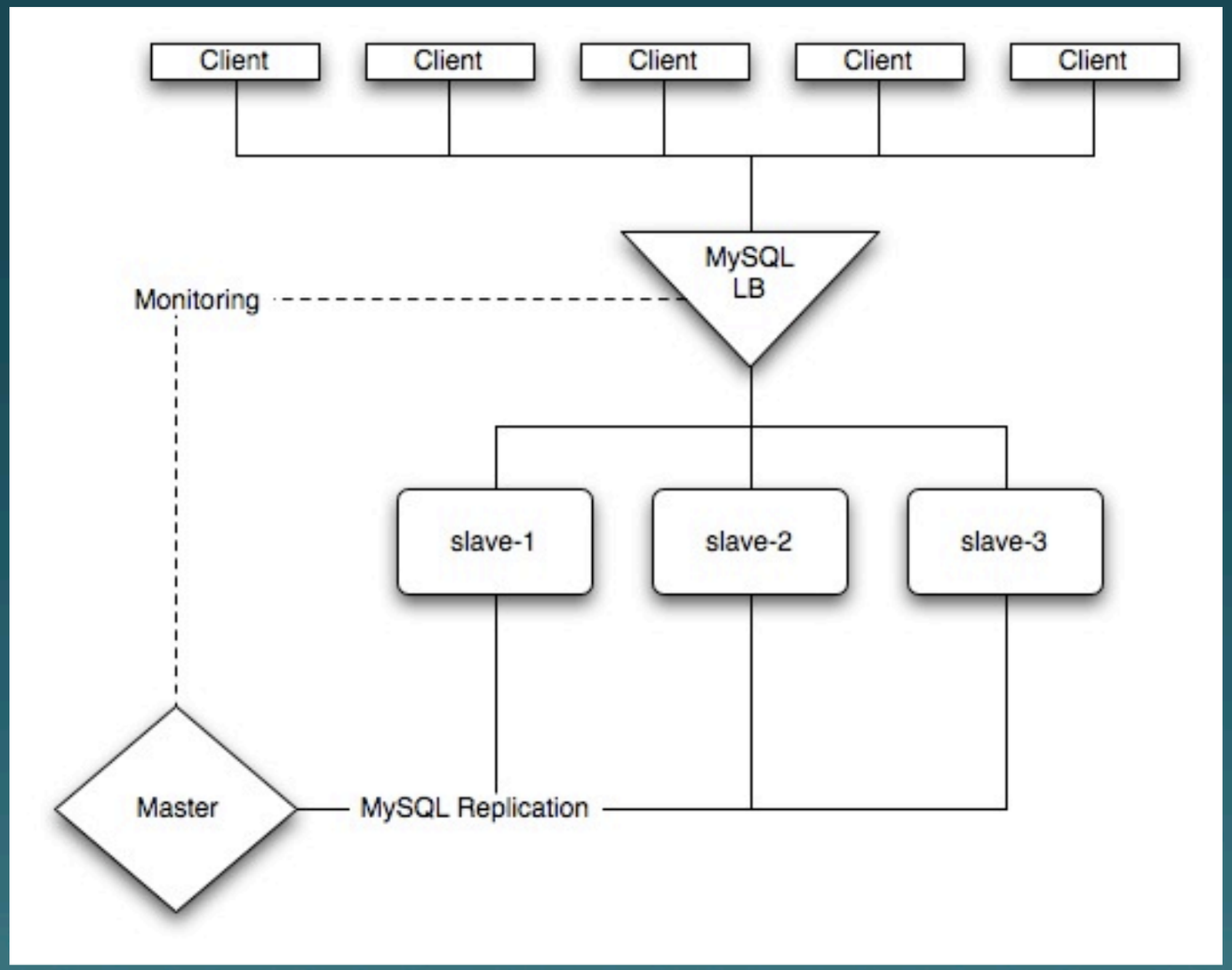
Topic:

Master Slave

The simple setup



The Advanced Setup



Example configuration

```php
<?php
class DATABASE_CONFIG {

    public $default = array(
        'datasource' => 'Database/Mysql',
        'persistent' => false,
        'host' => 'read-mysql.network.com',
        'login' => 'root',
        'password' => '',
        'database' => 'photoblog',
    );

    public $write = array(
        'datasource' => 'Database/Mysql',
        'persistent' => false,
        'host' => 'write-mysql.network.com',
        'login' => 'root',
        'password' => '',
        'database' => 'photoblog',
    );
}
```

Making it work

```
class AppModel extends Model {

    public function save($data, $validate = true, $whitelist = array()) {
        $old = $this->useDbConfig;
        $this->setDataSource('write');
        $result = parent::save($data, $validate, $whitelist);
        $this->setDataSource($old);
        return $result;
    }

    public function updateAll($fields, $conditions = true) {
        ...
    }

    public function delete($id = null, $cascade = true) {
        ...
    }

    public function deleteAll($conditions, $cascade = true, $callbacks = false) {
        ...
    }
```