Procrastinate!
Or at least your app should...

Sébastien Ballangé
Confoo 2024

ConFoo.CA
DEVELOPER CONFERENCE

CONNECT&GO

Who am I?

Staff developer at Connect&GO

15+ years of experience as a developer, mostly PHP and Symfony

CONNECT&GO

Introduction

- ➔ Users want responsive and fast applications
- ➔ Even when a lot happens behind the scenes

The main problem:
People are impatient but doing things right takes time

Let's ignore software for now

Software speed and performance is mostly a **perception** game.

Multiple options

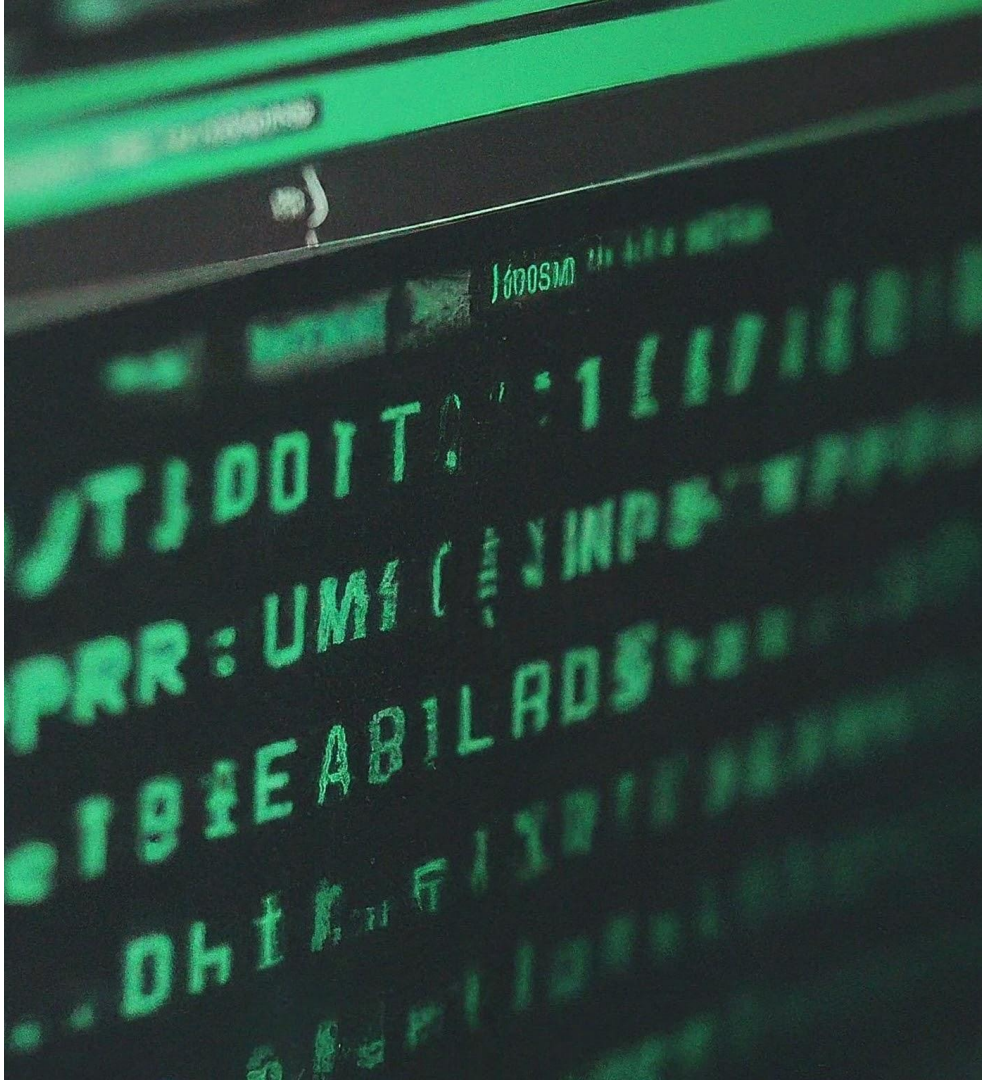
- ease of setup
- cost
- flexibility

Old school: cron jobs

- Can be set at any frequence (max once per minute)
- Relatively simple to set up
- Can be unnecessarily complex for some cases
- Concurrency can become an issue
- Watch out for time zones and daylight saving time

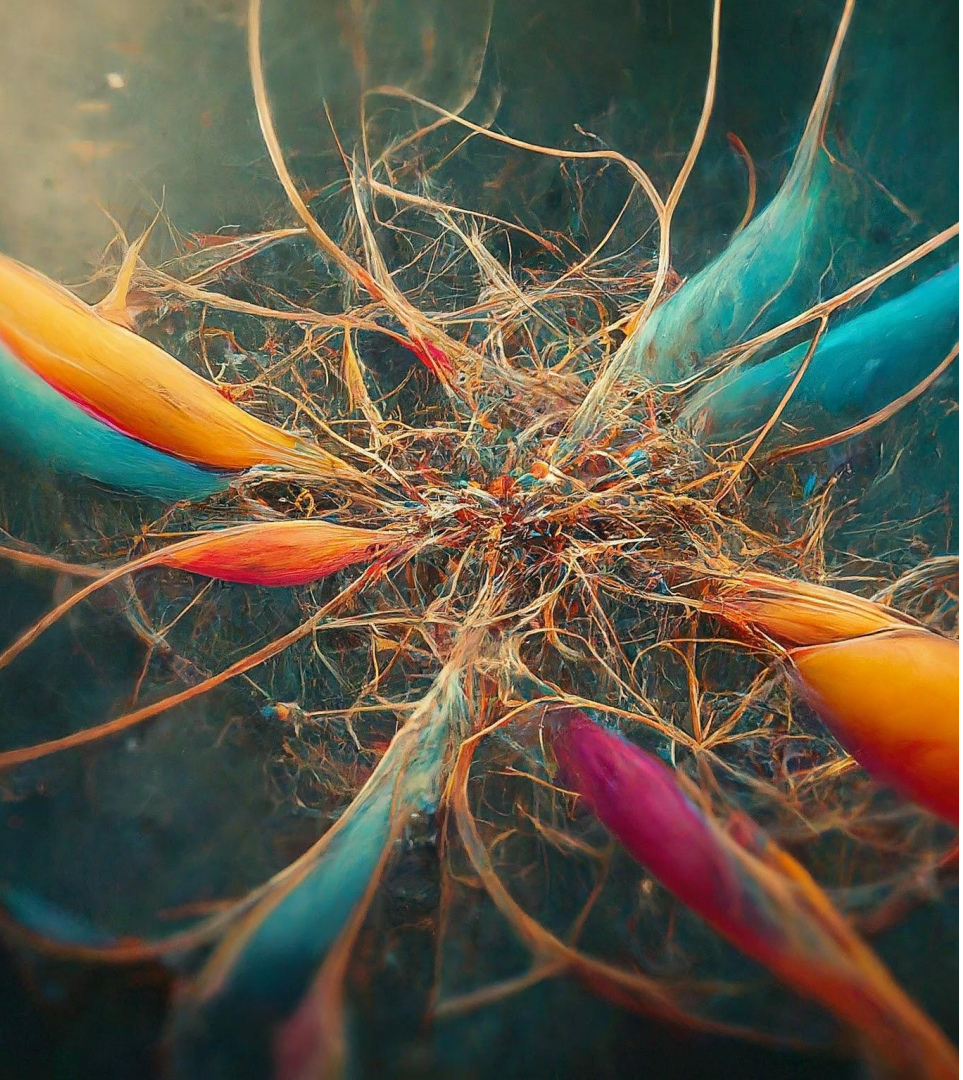
Shell exec

- Fire & forget
- Can run anything: an Artisan/Console command, call an API using cURL, …
- Hard to debug / troubleshoot
- Potential security risk



Example

```php
<?php

shell_exec("./bin/console my:command > /dev/null 2>&1 &");
```



Framework event system + fastcgi_finish_request()

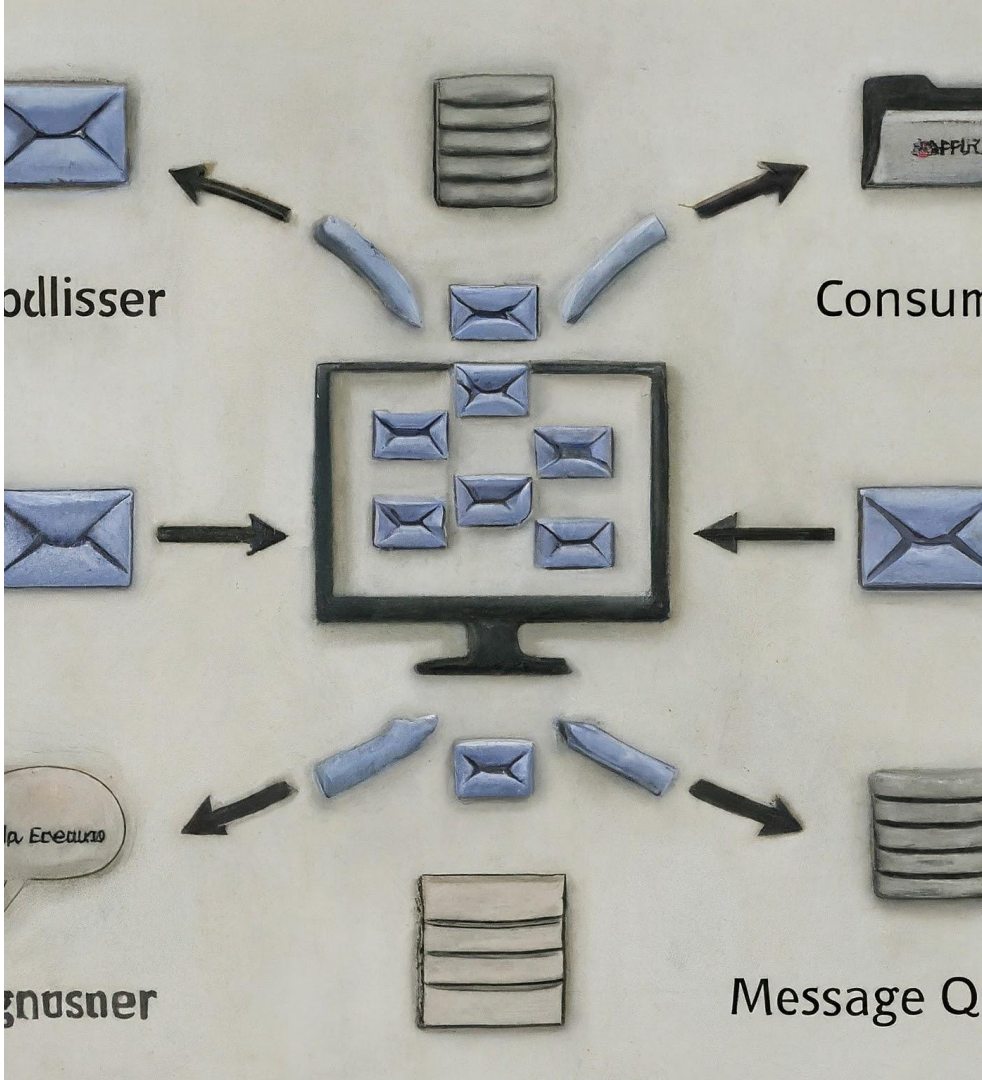
- Trigger an event anywhere in your code
- Process happens in the same request but after the page has been served
- Same codebase, nothing else to bootstrap: minimal overhead
- Prevent FPM from serving a new request until done
- Symfony: kernel.terminate Event
- Laravel: Terminable Middleware

Example - Symfony

```php
<?php

namespace App\EventListener;

use Symfony\Component\EventDispatcher\Attribute\AsEventListener;
use Symfony\Component\HttpKernel\Event\TerminateEvent;

#[AsEventListener]
final class MyListener
{
    public function __invoke(TerminateEvent $event): void
    {
        // do something potentially slow like sending an email
    }
}
```

Queuing systems

- RabbitMQ, Kafka are mature options
- Databases can also serve as a decent backend for low volume
- Scales horizontally
- Open the possibilities for more advanced flows
- Symfony Messenger
- Laravel Queues



Example - Symfony and Laravel

```php
<?php

namespace App\Message;

readonly class Notification
{
    public function __construct(public string $content, public string $recipient) {}
}
```

Symfony Messenger - handler

```php
<?php

namespace App\MessageHandler;

use App\Message\Notification;
use Symfony\Component\Messenger\Attribute\AsMessageHandler;
use Vonage\Client;
use Vonage\SMS\Message\SMS;

#[AsMessageHandler]
class NotificationHandler
{
    public function __construct(private Client $client, private string $sender) {}

    public function __invoke(Notification $notification): void
    {
        $sms = new SMS($notification->recipient, $this->sender, $notification->content);
        $this->client->sms()->send($sms);
    }
}
```

Symfony Messenger - dispatcher

```php
<?php

namespace App\Controller;

use App\Message\Notification;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\Messenger\MessageBusInterface;

class DefaultController extends AbstractController
{
   public function index(MessageBusInterface $bus): RedirectResponse
   {
       $message = new Notification('Hello World!', '+15145551234');
       $bus->dispatch($message);

       return $this->redirect('/home');
   }
}
```

Laravel Queues - handler

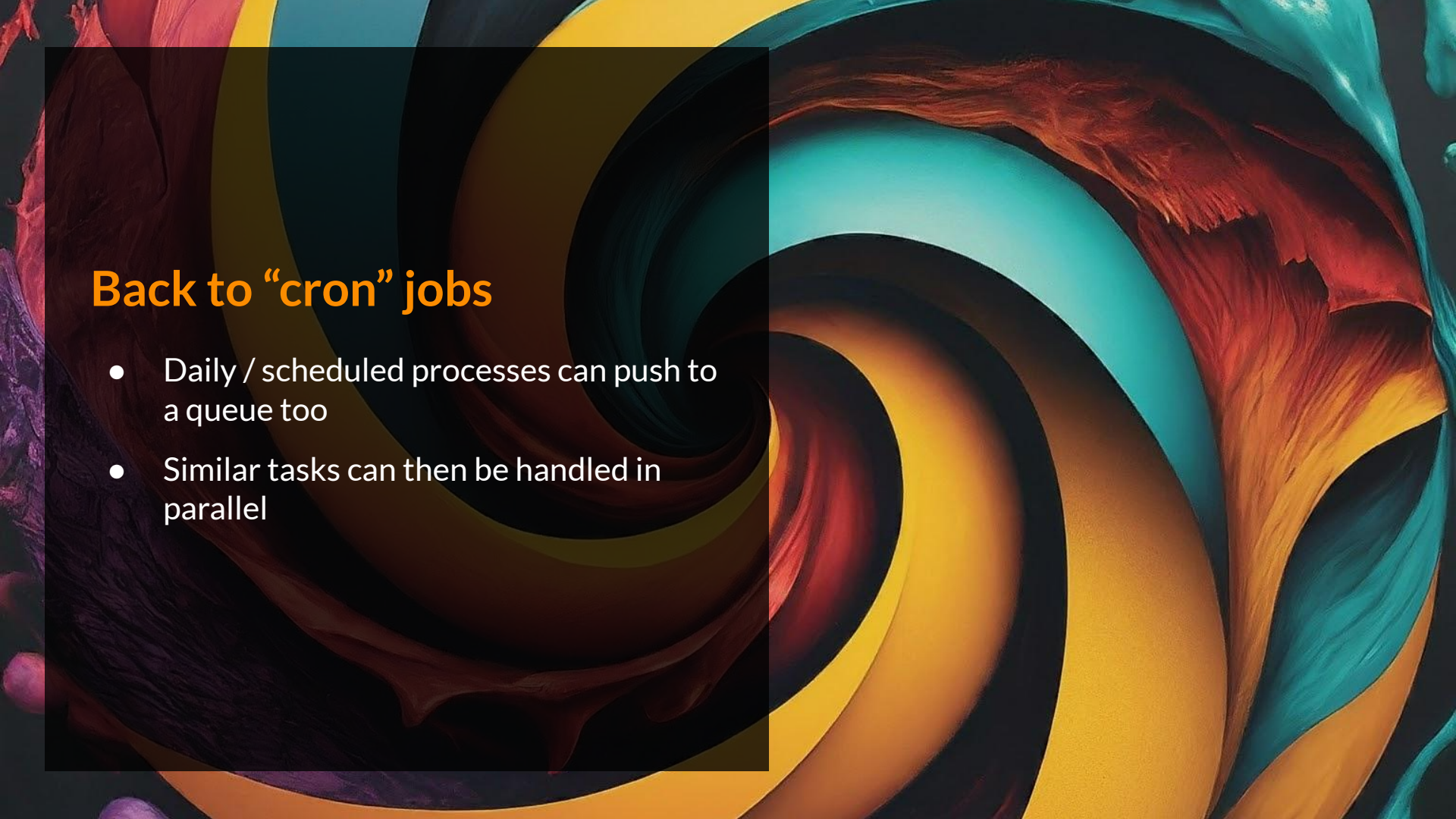
```php
<?php

namespace App\Jobs;

use App\Message\Notification;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Vonage\Client;
use Vonage\SMS\Message\SMS;

class SendNotification implements ShouldQueue
{
    use Dispatchable;

    public function __construct(public Notification $notification) {}

    public function handle(Client $client, string $sender): void
    {
        $sms = new SMS($this->notification->recipient, $sender, $this->notification->content);
        $client->sms()->send($sms);
    }
}
```

Laravel Queues - dispatcher

```php
<?php

namespace App\Http\Controllers;

use App\Jobs\SendNotification;
use App\Message\Notification;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;

class DefaultController extends Controller
{
   public function index(): RedirectResponse
   {
       $message = new Notification('Hello World!', '+15145551234');
       SendNotification::dispatch($message);

       return redirect('/home');
   }
}
```

Cloud variations

- AWS
  - SQS (queues)
  - SNS (notifications, pub/sub)
  - Event Bridge (serverless event bus)
- Azure
  - Service Bus
  - Event Grid
- Google Cloud's Pub Sub
- ...



Back to “cron” jobs

- Daily / scheduled processes can push to a queue too
- Similar tasks can then be handled in parallel

Conclusion

➔ **Background processing doesn't reduce the load on your infrastructure**

It's all about the **perceived** speed

➔ **Will help scaling in most cases**

Heavy calculation can be done outside peak hours

Allows to split a request over multiple servers

➔ **Make your app procrastinate!**

Thank you !

Find all slides from all presentations at
https://github.com/confooca/2024

ConFoo.CA
DEVELOPER CONFERENCE

CONNECT&GO