

**NOUVELLES
ARCHITECTURES**

Web temps Réel

**EST-CE QUE NODE.JS EST
FAIT POUR VOUS ?**

Auteur : David Négrier

TheCodingMachine™
TCM://

Introduction

On parle beaucoup de Node.js

Les discussions autour de Node.js sont de plus en plus intenses, même au sein de TheCodingMachine. Alors pourquoi Node.js fait-il tellement de buzz ?

Au fait, Node.js, c'est quoi ?

Node.js est un environnement (un toolkit) permettant de développer un serveur « Event-Driven » en JavaScript. En termes plus simples, cela signifie que les opérations de lecture ou d'écriture sont parallélisées contrairement au fonctionnement en mode thread où les opérations sont en série.

A quoi ça sert ?

Les applications qui nécessitent du temps réel (par exemple des cours de bourse ou un chat), des applications consommatrices en ressources telles que le streaming, l'upload de fichiers etc. De manière générale, toutes les applications où la vitesse d'exécution est un paramètre très important.

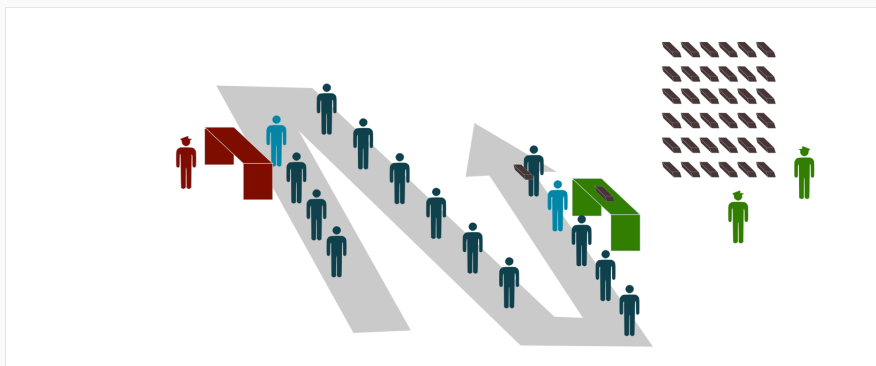
Comment ça marche ?

Comprendre le fonctionnement du Event-driven

Une image fréquemment employée pour décrire le mode de programmation « Event-driven » est celle d'une file de fast-food. Dans un modèle fonctionnant dans un mode Thread (la plupart des serveurs web actuels), le principe est d'attendre pour commander son menu puis d'attendre que la commande soit prête. La personne située derrière vous ne pourra pas passer sa commande tant que vous n'aurez pas reçu la votre.



Dans un mode « Event-driven », l'ordre est passé et vous attendez votre commande en dehors de la queue. Ainsi, la personne située derrière vous peut passer sa commande sans attendre.



Comment ça marche ?

Illustration de la cinématique Node.js

1. Votre navigateur fait une demande pour recevoir la « page1. html » au serveur Node.js
2. Le serveur Node accepte votre requête et appelle une fonction pour renvoyer la page
3. Tandis que le serveur Node attend la page (lecture disque), il procède au traitement de la requête suivante
4. Lorsque la page est retournée par la fonction, une fonction de callback est insérée dans la file d'attente du serveur Node
5. Le serveur Node exécute la fonction de callback qui, dans ce cas, renvoie la « page1.html » à votre navigateur.

Avantages

L'intérêt de cette nouvelle architecture ?

1. Elle est très rapide. Par exemple, une architecture Node.js/MongoDB permet d'envoyer 600 e-mails en 3 secondes tandis qu'une architecture classique PHP/MySQL le fait en 30 secondes.
2. La plupart des développeurs web connaissent JavaScript (au moins en surface). Aussi, trouver des compétences est très simple. Il n'y a que la notion de callback ou closure à comprendre pour pouvoir développer.
3. Elle permet d'uniformiser le langage de développement de toute la plateforme. Côté client (navigateur) et côté serveur.
4. Les fonctionnalités que permettent cette nouvelle architecture sont dans l'air du temps. Par exemple, les fils de discussions qui se mettent à jour automatiquement (réseaux sociaux), les outils de collaboration (chat, interactions sur un document etc.). Elle va dans le sens d'une expérience utilisateur de plus en plus riche en termes d'interactions, simple (donc éviter de changer de page par exemple) et dont la réponse est très rapide.
5. Flash est déclinant (au profit justement de JavaScript). Tandis que Node.js vient avec Socket.IO et permet donc de gérer de manière efficace le temps réel.
6. Parce que cette architecture est économe en termes de ressources matérielles.

... et puis, personne n'aurait parié sur le fait que JavaScript devienne un standard pour les navigateurs !

Inconvénients

Quelles sont les limites ?

Node.js a encore les défauts de sa jeunesse. Cette architecture ne propose que peu d'outils. Par exemple sur la gestion des tests, bibliothèques de fonction etc. Même si cette offre est en train de se constituer, il faudra du temps avant que n'émergent des standards.

Débugger des programmes écrits en Node.js peut être long et fastidieux. Au-delà, maintenir un programme en JavaScript est complexe car ce langage est très permissif aussi, la programmation doit faire preuve d'une grande rigueur.

Par exemple, si vous ne devez que servir des pages web avec des besoins en performance raisonnable, nous vous recommandons vivement de rester sur des architectures plus classiques (par exemple Apache/PHP/MySQL). Ces architectures proposent aussi des outils complets dans certains domaines. Par exemple, si vous souhaitez développer un site qui doit gérer beaucoup de contenus ou faire du e-Commerce, ne pas utiliser Drupal ou Magento serait dommage (pour ne pas dire plus).

Parce qu'il n'y a qu'un thread, une action en erreur peut avoir des répercussions sur tout le serveur. Contrairement à des langages comme PHP qui cloisonnent chaque requête dans un processus isolé.

Environnement Node.js

Qui utilise Node.js ?

Des entreprises telles que LinkedIn, Microsoft, Walmart ou encore eBay commencent à utiliser Node.js

A noter, une initiative intéressante de Yahoo! qui propose Cocktail, un environnement qui permet de développer des applications web sur différents types de plateformes, le tout basé en JavaScript (qui comprend Node.js).

<http://developer.yahoo.com/blogs/ymn/posts/2011/11/yahoo-announces-cocktails—shaken-not-stirred/>

La communauté

Une très forte communauté se développe autour de Node.js. D'ores et déjà, plus de 17.000 bibliothèques sont proposées - <http://npmjs.org> - et le gestionnaire de package, Node Package Manager, est très efficace.

Conclusion

Node.js a remis sur le devant de la scène l'architecture « Event-driven » et a prouvé que cette architecture était la plus efficace pour les performances. Cependant, parce que Node ne supporte que l'approche Event-driven et que celle-ci demande un effort de développement Node.js risque de se cantonner à son domaine de prédilection le web temps réel. Dans le futur, nous parions qu'une technologie regroupant le meilleur de l'Event-driven et du Thread verra le jour. Mongrel2, bien qu'incomplet, en montre la voie.

TheCodingMachine préconise d'utiliser Node.js **de manière tactique sur des architectures mixtes**. Selon les fonctionnalités que l'on souhaite développer, il suffit utiliser PHP pour les parties les plus classiques et Node.js pour les parties nécessitant du temps réel ou bien très consommatrices en termes de ressources.

En revanche, utiliser Node.js **pour migrer des applications de types desktop qui manipulent un grand nombre de données vers le web** (qui sont encore développées en swing dans certain grand groupe) peut avoir du sens.

Une tentative de prédiction (c'est souvent dangereux) pour terminer. Comme le flash est amené à moyen terme à disparaître et que le casual gaming reste une tendance forte. La technologie serveur Le Node.js serait-elle l'avenir de ces futurs jeux ?

Un projet ? Des idées ?
N'hésitez pas à nous contacter !

TheCodingMachine™
TCM://

www.thecodingmachine.com

contact@thecodingmachine.com

01 71 18 39 72



The licensor permits others to copy, distribute, display, and perform the work. In return, licenses must give the original author credit.

The licensor permits others to copy, distribute, display, and perform the work. In return, licenses may not use the work for commercial purposes -- unless they get the licensor's permission.