Coming next

Erlang

programmation

Mickaël Rémond

Ęřłąņġ

Remerciements à

Catherine Mathieu

ISBN : 2-212-11079-0

EYROLLES

Avant-propos

Écrire un livre sur Erlang n'est pas chose aisée et laisse forcément un sentiment d'inachevé : le langage est si puissant, l'environnement de développement contient des fonctionnalités si nombreuses et les bibliothèques sont si riches, qu'il s'avère impossible de couvrir, de manière détaillée, en un seul ouvrage, l'ensemble des possibilités offertes par ce langage. Pour mémoire, la documentation officielle, en anglais, occupe une étagère complète de bibliothèque.

C'est bien là le paradoxe. Erlang est un des langages les plus puissants aujourd'hui disponibles. Il est utilisé avec succès dans d'importants projets industriels, il est abondamment documenté, mais reste aujourd'hui encore méconnu. Le débutant souffre vraisemblablement d'une abondance de documentation très technique, parfois imprégnée de la « culture télécoms » des créateurs du langage.

Ce livre tente d'y remédier. Il offre au développeur désireux de se lancer dans l'apprentissage d'Erlang un outil pour comprendre ce qui fait l'essence et la particularité de ce langage.

Le code source des études de cas est disponible en téléchargement sur le site d'accompagnement à l'adresse www.editions-eyrolles.com.

À qui s'adresse ce livre ?

Les points forts d'Erlang sont assurément orientés réseau. Toute personne impliquée dans la réalisation d'applications « serveur » robustes, tolérantes aux pannes et capables de monter en charge, qu'il s'agisse de développeurs ou de chefs de projets, trouvera un intérêt dans la lecture de cet ouvrage.

Plus généralement, Erlang bouleverse la manière d'appréhender le développement. Nombre de ceux qui l'ont essayé affirment aujourd'hui que leur manière d'aborder un problème informatique s'en trouve modifiée, y compris au cours de développements réalisés dans des langages plus traditionnels. Gageons que la connaissance de ce langage devra faire partie de la culture d'un bon développeur !

Bien entendu, cet ouvrage s'adresse également aux administrateurs système qui gèrent quotidiennement des applications Erlang dans leur environnement de production. Il leur donne les principes nécessaires à la bonne gestion d'applications Erlang.

Pourquoi Erlang aujourd'hui ?

La programmation concurrente pour un modèle de développement orienté services

Sous l'influence des transformations liées à l'essor de l'Internet, l'informatique s'oriente aujourd'hui vers un modèle à base de services. Ce modèle n'est ni plus ni moins qu'une distribution fonctionnelle décentralisée des systèmes informatiques. Les protocoles SOAP et WSDL apportent une réponse pratique à la problématique des services Web. Ils ne sont cependant que la partie émergée de l'iceberg. La vrai question reste : comment concevoir et mettre en œuvre des systèmes distribués robustes ? Quels sont les mécanismes sous-jacents ?

En effet, le modèle des services Web s'appuie sur une approche plus stratégique afin de remplir un objectif fonctionnel : les systèmes informatiques ne doivent plus être cantonnés aux limites de l'entreprise ; ils doivent être capables de collaborer entre eux, chacun détenant une petite partie de l'application globale. Fort de ce constat, le concept de services Web en vient à proposer un système qui repose sur le passage de messages, fort similaire à celui sur lequel Erlang repose.

Nous assistons donc à une convergence vers un modèle fonctionnel distribué, à base de services faiblement couplés, qui tend à remplacer le modèle d'interopérabilité à base d'objets, fortement couplé comme CORBA (*Common Object Request Broker Architecture*), pour la construction de systèmes d'information. Cela signifie que la dépendance entre tous les éléments du système doit être relativement faible pour tolérer des coupures réseau, voire ne reposer sur aucune connexion permanente. Le couplage doit être faible également dans la conception des divers éléments de l'application : chacune des applications du système doit pouvoir être développée indépendamment des autres, tout en étant capable de s'intégrer dans le système d'ensemble. C'est cela que permet de mettre en œuvre l'approche du développement par service.

S'appuyer sur les acquis du langage Erlang pour réaliser de tels systèmes distribués orientés services permet de disposer d'une avance considérable. Erlang est le langage de choix pour développer dans ce type d'architecture.

Le mode de fonctionnement d'Erlang correspond, de par sa conception même, à celui qui est aujourd'hui promu par l'avènement des services Web. Conception, architecture, mécanismes de tolérance aux pannes… Erlang apporte une réponse générale à la problématique du développement orienté services.

Il est ainsi traditionnel en Erlang de créer des ensembles applicatifs dont les éléments fonctionnent de manière concurrente. Ces programmes partagent des informations en se synchronisant par un échange de messages. L'ensemble constitue une application distribuée, dont l'organisation physique importe peu : que les programmes tournent localement ou que certains programmes soient présents sur telle autre machine n'a aucune incidence.

Erlang et les services Web : les vertus de l'approche hybride

Actuellement, le développement d'architectures orientées services connaît deux faiblesses majeures, à savoir les performances et la robustesse. Une approche « hybride » faisant intervenir Erlang serait idéale :

- **Performances.** Le principal inconvénient des services Web est le coût de traitement induit par l'introduction d'un protocole tel que SOAP dans les échanges entre programmes. Le temps

nécessaire à la lecture du flux XML et à sa transformation dans une structure de données propre au programme qui souhaite l'exploiter peut considérablement nuire à son utilisation. Cette lacune fait qu'il est préférable de limiter l'utilisation de SOAP au cas où l'interopérabilité est absolument nécessaire et où SOAP est le seul moyen d'assurer cette interopérabilité. Si l'on échappe à ce cas de figure, il est préférable d'utiliser un autre point d'entrée que l'interface SOAP. Cela signifie également que, lors de la construction d'un système, SOAP ne doit pas être utilisé pour faire communiquer chacun des programmes du système, mais uniquement pour implanter la communication du système avec les programmes extérieurs.

Pour les communications internes, entre les éléments du système, il est préférable d'utiliser le mécanisme standard d'Erlang pour la communication entre nœuds.

- **Robustesse et maturité.** Les services Web gagneraient à intégrer les mécanismes d'Erlang permettant de rendre une application distribuée robuste, tolérante aux pannes, etc. Erlang implémente un modèle élaboré depuis une quinzaine d'années, qui a fait ses preuves sur des applications industrielles.

Le modèle de développement d'une application à base de services Web étant proche du mode de développement des interfaces Erlang, il est très simple de doter un programme Erlang existant d'une interface SOAP. Nous bénéficions ainsi de tous les avantages relatifs à la construction d'une application OTP et d'une ouverture sur l'extérieur extrêmement simple à obtenir. Nous pouvons ainsi réunir le meilleur des deux mondes !

Erlang, langage « glu » et middleware

Pour qualifier un langage, il est courant d'entendre le terme de « langage glu ». Cette définition insiste sur le rôle pivot que joue un langage pour fédérer des développements hétérogènes. Erlang peut être défini comme tel du fait de sa capacité à interopérer avec les autres langages.

À la différence des langages de script habituellement qualifiés de langages glu, Erlang ne se contente pas de fédérer des applications : il les fédère aussi au niveau d'un réseau, lorsqu'elles sont réparties sur un ensemble de machines. À ce titre, l'environnement de développement Erlang mérite plus le nom de logiciel médiateur, ou *middleware*. Un développeur habile est même capable de transmettre les caractéristiques de ce nouvel ensemble au développement résultant. Erlang est ainsi souvent utilisé pour transformer des systèmes classiques en outils robustes, tolérants aux pannes, capables de monter en charge.

Le terme de glu pourrait aussi laisser croire qu'Erlang n'est qu'un langage de script peu adapté aux développements de grande envergure. C'est tout le contraire. Une application Erlang comprend souvent près de quatre fois moins de lignes de code – étant admis qu'il y a le même nombre de bogues dans une ligne de code Erlang que dans une ligne d'un autre langage. On en déduit que la productivité des développeurs est considérablement accrue, et que les avantages à développer en Erlang se font d'autant mieux sentir que le projet est de taille importante. De fait, Erlang a déjà été utilisé pour réaliser des développements d'une taille considérable. C'est lui qui a été choisi pour implémenter le routeur AXD 310 d'Ericsson. Dans l'échelle des créations humaines, un tel routeur se classe à un niveau de complexité supérieur à un satellite artificiel.

Questions fréquentes sur Erlang

Pourquoi utiliser Erlang ?

Plusieurs raisons peuvent justifier l'utilisation d'Erlang :

- **Développements orientés serveur.** Les développements orientés serveur impliquent quasi nécessairement la gestion de la concurrence. Les serveurs ont pour objet de servir de ressources à des clients accédant au programme de manière simultanée. Erlang excelle pour tous ces développements.
- **Développements orientés réseau.** Si l'on souhaite développer un programme mettant en œuvre des ressources réseaux, Erlang est un langage qui s'impose rapidement. C'est un middleware très complet. Les outils pour les développements orientés réseaux sont largement disponibles en Erlang.
- **Qualité du code.** Pour développer un programme tirant partie des principales caractéristiques du langage, l'environnement de développement Erlang devient un outil précieux : migration de code pour l'organisation de la haute disponibilité, mise à jour du code à chaud, réplication des données, etc. Erlang met déjà en œuvre ces mécanismes.
- **Productivité.** Sur des projets d'envergure, la productivité des développeurs entre pour bonne part dans le coût du projet. Erlang permet d'accélérer les développements et de réduire les coûts.

Erlang est-il un langage ou un environnement de développement ?

Les deux. Erlang est constitué de plusieurs éléments et se rapproche davantage de l'environnement de développement que du simple langage. En outre, il joue par certaines fonctions le rôle de système d'exploitation (gestion des processus, ordonnancement de l'exécution des tâches, etc.).

Erlang est-il uniquement adapté au domaine des télécommunications ?

Erlang est issu du secteur des télécommunications. Sa principale caractéristique est la concurrence. Les applications de télécommunications sont loin d'être les seules à pouvoir être qualifiées de concurrentes. La concurrence d'Erlang trouve à s'appliquer dans de nombreux autres types d'applications :

- client/serveur traditionnel ;
- développement de « services » ;
- systèmes « pair à pair » ;
- interfaces utilisateur.

Pour montrer la portée généraliste d'Erlang, citons Wings3D, un modeleur 3D développé en Erlang, **Er**macs, une version du célèbre éditeur de texte Emacs, basée sur Erlang plutôt que sur Lisp, etc.

Erlang est-il très compliqué à apprendre ?

Erlang est un langage extrêmement simple à apprendre. Sa syntaxe est limitée et s'assimile très vite. Les débutants deviennent très vite productifs dans ce langage dès lors qu'ils acquièrent les bases nécessaires et la philosophie du langage.

Erlang permet de rendre simples les développements les plus complexes. La programmation concurrente, dite « multithreadée » constitue bien souvent un horizon difficile à atteindre pour le programmeur

débutant ou occasionnel. En Erlang, les développements concurrents constituent une approche naturelle du développement. Les débutants sont capables de réaliser des programmes composés de plusieurs processus (*threads*) dès leur premier jour d'apprentissage du langage.

Le langage C est-il seul à permettre la réalisation d'applications serveur performantes ?

L'idée que seul le langage C permet la réalisation d'applications serveur performantes est couramment répandue. Apache, par exemple, l'une des applications serveur les plus utilisées dans le monde, est développé en langage C.

Or, la réalisation d'un serveur Erlang permet d'obtenir des performances comparables à ce qu'il est possible d'obtenir avec d'autres langages comme le C.

> Il existe une application Erlang, baptisée Yaws qui fournit des performances similaires à celles d'Apache pour servir des pages HTML statiques. Le modèle de processus léger d'Erlang permet à Yaws de supporter la montée en charge mieux qu'Apache. Pour finir, le modèle de développement dynamique d'Erlang lui permet d'offrir des performances supérieures au couple Apache-PHP pour servir des pages dynamiques, grâce à une intégration étroite du moteur de génération de pages au cœur du serveur Web Yaws. Les performances sont similaires à la réalisation d'un site Web dynamique pour Apache par la création de modules Apache, mais le développement en est beaucoup moins complexe.

Erlang fait le choix d'un modèle de développement fonctionnel par rapport à un modèle de développement objet pour des raisons de performance et de constance dans les temps de réponse de l'application. La superposition des couches objet à travers le mécanisme d'héritage peut pénaliser les temps de réponse. Or Erlang est conçu pour être un langage temps réel mou. Il doit permettre d'offrir des temps de latence très faibles pour démarrer le traitement de nouvelles opérations. Pour offrir ce type de garanties, le modèle objet n'a pu être retenu.

Erlang est donc une alternative crédible pour la réalisation de serveurs haute performance, hautement disponibles.

Erlang n'est pas un langage objet : n'est-il pas obsolète ?

Erlang propose les avantages offerts par les langages fonctionnels aussi bien que les avantages offerts par les langages objet :

- **Développement algorithmique.** Le caractère fonctionnel du langage en fait un langage adapté pour l'implémentation des algorithmes. Le développement des algorithmes est toujours traité avec une approche fonctionnelle, y compris dans les langages de développement orientés objet. Disposer d'un langage fonctionnel très puissant permet de développer plus rapidement et plus simplement des algorithmes complexes.
- **Conception.** Le développement orienté objet relève d'une approche de conception des programmes et non de développement d'algorithmes. Erlang dispose du framework OTP qui met en œuvre une technique de conception des programmes Erlang au moins aussi puissante que l'approche objet. Le framework OTP est une approche du développement basé sur les motifs récurrents de programmation

en langage Erlang. Tout comme la conception dans l'approche objet, le framework Erlang/OTP s'appuie sur des modèles de conception (*design patterns*).

- **Architecture.** L'environnement Erlang en plaçant au centre de son mode de développement la concurrence et la distribution des traitements est un outil middleware précieux pour les architectes des systèmes d'information, y compris pour assembler des développements effectués dans d'autres langages.

L'avenir du langage semble assuré par la nécessité d'intégrer robustesse et haute disponibilité dans les développements stratégiques. Par ailleurs, l'avènement des architectures orientées service conduit vers une approche fonctionnelle du développement.

Erlang excelle dans les domaines qui ont trait aux réseaux et à la programmation de systèmes robustes. Son principal inconvénient reste pour l'instant sa relative confidentialité. Opter pour Erlang est aussi une question de *feeling* : si ce choix dépend de l'adéquation d'Erlang à un besoin donné, il dépend aussi de la facilité avec laquelle le développeur peut modéliser un problème en Erlang, qui reste un langage doté d'une certaine « personnalité ». Mais, une fois le paradigme Erlang assimilé, le sentiment de simplification de problèmes complexes qu'il procure est tel que le développeur risque fort de ne plus vouloir s'en passer.