



Préface

Parler d'optimisation logicielle, c'est revenir à un fondamental de la programmation que de nombreux développeurs oublient. Rappelons que les 640 Ko de mémoire des premiers processeurs x86 ou les espaces mémoire réduits des systèmes embarqués ont conduit les programmeurs à faire des prodiges !

Si les ressources sont devenues facilement accessibles et nombreuses, elles ont pourtant bel et bien un coût. Certes le prix d'achat du matériel ne fait pas resurgir ce coût réel, mais il est possible de chiffrer la consommation énergétique d'un code en exploitation, et de trouver une corrélation entre les ressources utilisées et leur équivalent énergétique, voire équivalent carbone.

L'optimisation d'un code va tenter de réduire le jeu de ressources utilisées et cela va se traduire par une vitesse accrue d'exécution. Cette variation sera plus ou moins perceptible par l'utilisateur devant sa machine. L'impact est encore plus marqué dans une ferme de serveurs. Si un code est optimisé de 20 %, cela se traduit sur un ordinateur portable de manière à peine sensible (meilleure réactivité, voire allongement de la durée de vie d'une charge de batterie). Par contre, dans une ferme de serveurs, 20 % représentent directement une machine sur cinq en moins ! Rappelons l'expérience de Facebook qui, en générant du code C++ à partir du PHP de leur page (projet HipHop), a obtenu un gain moyen de 50 %. Il faut donc deux fois moins de serveurs à Facebook pour fonctionner, soit 60 000 tonnes de CO₂ économisées. Là, ça devient plus que significatif !

Écrire du code .NET performant

Profilage et bonnes pratiques

L'optimisation, c'est aussi prendre en compte l'architecture des nouvelles machines. Nous le voyons depuis quelques années, la loi de Moore continue, mais en multipliant les cœurs et non la fréquence horloge qui atteint des limites physiques. Pour être efficaces, nous devons donc aujourd'hui produire du code parallèle sachant exploiter des architectures multicœurs, hyperthreadées... Ces codes parallèles vont avoir tendance à consommer plus d'énergie mais dans un laps de temps plus court que leur équivalent monthread. Le gain peut, là aussi, facilement atteindre les 30 à 50 % et devenir un véritable gisement d'économies.

L'émergence du cloud computing et des datacenters géants des grands acteurs va mettre en exergue que la performance n'est pas une option ou un caprice intellectuel, mais se traduit par un impact économique et environnemental très fort. La rentabilité de ces datacenters est directement liée à la qualité et à la performance du code qui s'y exécute. Enjeux fondamentaux pour les développeurs et notre industrie.

L'optimisation du code n'est plus une option au regard d'une planète aux ressources finies.

Au-delà des choix et solutions que mettront en place les grands acteurs mondiaux, l'écriture d'un code performant est à la portée de tous les développeurs. Comme pour tous les éco-gestes que les citoyens lambda que nous sommes sont invités à adopter, le développeur doit aujourd'hui se préoccuper de produire un code performant, moins gourmand en ressources. Il n'a pas besoin d'être un expert dans son langage pour y parvenir, c'est souvent une affaire de bon sens et de bons réflexes et Jean-Philippe Gouigoux le démontre dans ce livre. Je suis sûr que le lecteur y trouvera des clés efficaces pour coder autrement et s'engager sur la voie du Développement Durable.

Eric Mittlette - Microsoft France

Responsable de l'équipe relation technique avec les développeurs

Responsable du Développement Durable chez Microsoft France