Si on ne peut pas changer le regard du monde académique sur la programmation, que faire ?

Daniel Porumbel[1]

CEDRIC, CNAM, Paris, France daniel.porumbel@cnam.fr

> Beaucoup de livres servent simplement à montrer combien il y a de faux sentiers et comme on peut sérieusement s'égarer, si on les suit. Mais celui qui pense par lui-même possède la boussole qui lui fera trouver le vrai chemin. Chasser ses pensées originales pour prendre un livre en main, c'est un péché contre le Saint-Esprit [ma note : suivre un tel ou un tel sur Twitter en 2023 n'est pas mieux].
>
> *Arthur Schopenhauer*

1 Introduction, débat et conclusion

Soit les deux programmes C++ ci-dessous (que le main). Les deux sont exponentiels en théorie, mais la réalité pratique est plus imprévisible : le programme à gauche est exponentiel alors celui de droite est *de facto linéaire* pour Monsieur-Tout-le-Monde dans la vie de tous les jours (voir sondage ci-dessous). La valeur seuil de n pour laquelle l'allocation mémoire échoue et le programme ne peut pas tourner est la même dans les deux cas. Une vidéo qui montre le phénomène est disponible à www.youtube.com/watch?v=9NE0rdLM3Zg. J'ai déjà présenté ces deux programmes l'an dernier dans le cadre de la même session. Je renvoie le lecteur vers ma communication de 2022 pour des explications techniques. [1] Mais beaucoup ont pu penser à l'époque que tout cela n'était qu'un cas isolé lié à une configuration atypique de mon Linux.

Or j'ai depuis vérifié ce comportement avec mes collègues du laboratoire et aussi via un sondage public sur la liste du GDR RO le 21/11/22. Environ 50 personnes m'ont confirmé sur leurs machines que le programme à droite s'exécute en un temps linéaire. Une seule personne a eu une machine (virtuelle) qui demande un temps exponentiel. Une chose est claire : s'il y a une machine avec une configuration atypique, ce n'est pas une de ces 50 machines suscitées qui ont confirmé le comportement, mais celle qui l'a infirmé. Soit un programme de complexité linéaire prouvée théoriquement. Est-ce que c'est garanti qu'il va prendre un temps linéaire si on l'exécute sur 50 machines ? Si vous mettez votre tête à couper que c'est le cas, ça me paraît très risqué de votre part. C'est propre à **la nature humaine : les certitudes n'existent qu'en théorie.**

```
int n = 30;
int z = 1<<n;            //2^n
z++;
int*tab=(int*)malloc(z*sizeof(int));
for(int i=0;i<z;i++)
    tab[i] = 0;
for(int i=0;i<n;i++)
   if(tab[1<<(i+1)] < 7 + tab[1<<i])
      tab[1<<(i+1)] = 7 + tab[1<<i];
cout<<tab[z-1]<<endl; //toujours 7*n
```

```
int n = 30;
int z = 1<<n;            //2^n
z++;
int*tab=(int*)calloc(z,sizeof(int));
//recall calloc returns by default
//a memory block filled with zeros
for(int i=0;i<n;i++)
   if(tab[1<<(i+1)] < 7 + tab[1<<i])
      tab[1<<(i+1)] = 7 + tab[1<<i];
cout<<tab[z-1]<<endl; //toujours 7*n
```

Cet exemple suffit pour montrer les limités de la notion de complexité théorique classique sur les machines des plus communs des mortels. Monsieur-Tout-le-Monde est obligé de séparer les dures réalités de la vie réelle des enseignements académiques qu'il a pu suivre sur les bancs des écoles. Car l'efficacité pratique d'un bloc de code est toujours plus imprévisible qu'on ne le croit. En fait, tout programmeur-chercheur chevronné en RO l'a déjà constaté mille fois ; les programmes ci-dessus sont juste un petit exemple.

1. Une réponse à mon message sur la liste du GDR RO explique que si on utilise l'option d'optimisation -O3, g++ remplace tout seul le programme à gauche avec celui de droite. Donc les deux programmes peuvent avoir de facto une complexité linéaire en contradiction avec la complexité théorique. Mais je ne retrouve pas cette complexité linéaire en utilisant la méthode C++ d'initialisation, c.à.d., new int[z]().

J'insiste que ceci est un aspect souvent oublié en informatique : il existe une (fréquente) décorrélation entre le temps de calcul théorique et le temps de calcul réellement observable en pratique. Lors de mon sondage sur la liste du gdrro, plusieurs personnes ont commencé à donner des éléments sur `calloc` et `malloc` pour expliquer cette décorrélation sur les deux programmes évoqués ici. Mais ce qui devait arriver est arrivé : quelqu'un a rebondit pour exprimer ses doutes que cela soit vraiment un sujet de recherche qui mérite d'être débattu sur la liste. La discussion a été arrêtée. *Cela rejoint la question du titre.*

Est-il possible de concilier le monde académique et la programmation pratique ? La réponse me paraît négative, au delà de quelques généralités. Trop de chercheurs académiques pensent que la programmation n'est pas assez noble pour s'y intéresser en profondeur ; beaucoup pensent que « c'est de la cuisine ; on s'arrange. Il faut en faire, mais peu importe le langage choisi ; de toute façon, quelle grande différence entre Java, C/C++, Fortran, ou je-ne-sait-quoi ? ». Par exemple, quelqu'un m'a même dit une fois qu'on peut coder tous les algorithmes du monde en Fortran. J'ai trouvé la remarque tout à fait légitime et ma réponse fut « c'est vrai ». Mais j'ai réalisé plus tard que cela portait à croire qu'on peut légitimement ignorer tout ce qu'on a inventé en informatique depuis les années 1950. Mais le manque de corrélation entre le temps de calcul théorique et le temps de calcul pratique (dont on n'a donné qu'un seul exemple plus haut) s'explique par des questions de compilation ou de systèmes d'exploitation bien plus récentes que 1950.

Comment sensibiliser les académiques qu'on a besoin d'une notion de complexité pour les plus communs des mortels à côté de la complexité théorique ? Je pense que la décorrélation entre les deux augmente décennie après décennie. J'essaye de décrire le blocage : toutes les idées discutées publiquement (qui touchent ce sujet) sont souvent très arides, au mieux strictement techniques. Mais les plus graves déficiences de nos communautés ne sont pas seulement d'une nature technique à décrire avec des équations.

Même dans un monde idéal, on est obligé d'avoir un cadre institutionnel qui génère un équilibre de forces qui implique souvent une forme de hiérarchie (implicite). Les chercheurs débutants vont toujours prendre les codes (sociaux) des chercheurs confirmés ; la petite bourgeoisie va toujours copier la grande bourgeoise ; la classe ouvrière va s'accorder avec la classe moyenne, et ainsi de suite, couche par couche.

La preuve par autorité peut fonctionner aussi bien en science que dans tout autre secteur de la vie. Il existe une fiche wikipedia en anglais « Argument from authority » qui dit : « leading American zoologist Th. Painter declared, based on poor data and conflicting observations he had made, that humans had 24 pairs of chromosomes. From the 1920s until 1956, scientists propagated this "fact" based on Painter's authority, despite subsequent counts totaling the correct number of 23. Even textbooks with photos showing 23 pairs incorrectly declared the number to be 24 based on the authority of the then-consensus of 24 pairs. »

Petit à petit cela conduit à une situation où on se soumet de facto à une majorité (très) influencée par quelques sommités académiques. L'influence du haut vers le bas peut rester très forte, bien qu'elle soit exercée très discrètement, par diffusion subtile couche par couche en cascade. Il n'y a pas de sortie sauf peut-être une forme d'activité sous anonymat (ou clandestine) qui est hors de question dans le cadre de cette modeste session – car même une communauté sous anonymat a besoin de structure pour fonctionner.

Dans l'esprit de la devise de cette communication, il est possible de rompre avec cette dynamique (uniquement) au niveau personnel. Mais cela est plus difficile : l'homme doit beaucoup travailler tout seul pour se faire une idée exacte de ce qui se passe dans son domaine. Le « prêt à penser » servi par les livres ou par d'autres sources extérieures n'est pas suffisant pour pénétrer durablement l'essence des choses. Certains vont beaucoup hésiter avant de concevoir dans leur tête une seule idée qui n'est pas déjà présente dans la tête d'autres gens (plus haut placés). C'est une tendance naturelle parfois utile (par exemple, dans une démarche d'intégration), mais on doit faire attention à un autre piège : « un homme dans cette situation est assailli, à chaque pensée qui surgit dans sa tête, par le souci de savoir si elle s'accommode avec les vues de l'autorité supérieure. Cela paralyse son esprit, au point que les pensées mêmes n'osent plus surgir »

Je viens de citer « La Philosophie universitaire » de A Schopenhauer, un essai avec d'innombrables retombées sur la science moderne. Il conclut que il est impossible de suivre fidèlement la voie de la vérité pour trop longtemps dans un cadre institutionnel : « S'il ne s'agissait donc dans l'affaire que de progrès de la philosophie et de l'avancement dans la voie de la vérité, je recommanderais, comme la meilleure chose, d'arrêter le combat simulé qui se livre à ce sujet dans les universités. Car celles-ci ne sont vraiment pas le lieu favorable pour une philosophie sérieuse et sincère, laquelle doit y céder trop souvent la place à une marionnette qui lui a pris ses vêtements, et se voit contrainte de parader. » Les Dieux ont placé la science sous une malédiction : pour la cultiver, on doit bâtir des institutions mais dès qu'on bâtit des institutions, on n'est plus dans un milieu favorable et sincère ; le charme se rompt progressivement et la dynamique générale est vouée à devenir rigide, froide, artificielle. J'espère avoir donné quelques pistes pour s'y extraire. Le but n'est pas de se révolter, mais de forger ou de préserver une autre conception du rapport des sciences du logiciel à la théorie.