MLKAPS: Machine Learning for Kernel Accuracy and Performance Studies

Mathys Eliott Jam[1] Eric Petit[2] Pablo Oliveira[1] David Defour[3] Greg Henry[2] William Jalby[1]

[1]LI-PARAD
Université Versailles St-Quentin en Yvelines (UVSQ)

[2]Intel Corp.

[3]Université Perpignan Via Domitia

June 26, 2025

Introduction

Exemple Jouet

```
1 void kernel(double* data, int m, int n, int nthreads)
2 {
3     #pragma omp parallel num_threads(nthreads)
4     ...
5 }
```

L'utilisateur doit fournir tout les paramètres au noyaux. On distingue :

- **Les paramètres d'entrées (data, m, n)**
- **Les paramètres de designs (nthreads)**

Exemple Jouet

```
void kernel_wrapped(double* data, int m, int n)
{
    int nthreads = dt_nthreads(data, m, n);
    kernel(data, m, n , nthreads);
}
```

On cherche un mécanisme de décision runtime - ici `dt_nthreads(...)` - pour trouver automatiquement les **paramètres de designs** selon les **paramètres d'entrées**.

Exemple Jouet

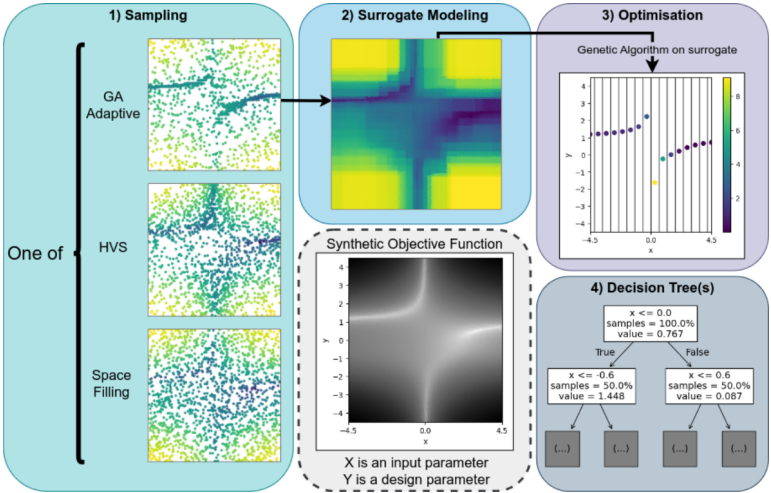
```
1 void kernel_wrapped(double* data, int m, int n)
2 {
3     int nthreads = dt_nthreads(data, m, n);
4     kernel(data, m, n , nthreads);
5 }
```

On cherche un mécanisme de décision runtime - ici `dt_nthreads(...)` - pour trouver automatiquement les **paramètres de designs** selon les **paramètres d'entrées**.

Soit O une fonction objective du noyau. On cherche une fonction $DT : I \to D$ telle que, pour chaque $i_k \in I$:

$$DT(i_k) = d_k = \arg\min_{d \in D} O(i_k, d)$$

Workflow de MLKAPS



Résultats

Environnement

	CPU	freq.	cores	threads	L1	L2	L3	RAM	Type
KNM	Intel Knights Mill	1.5GHz	72	288	32 KB	36 MB	—	16 GB	HBM
SPR	Intel Xeon Gold 6438M	2.2GHz	64	128	80 KB	2 MB	60 MB	504 GB	DDR5

On s'intéresse aux noyau `dgetrf` (LU) de la Intel MKL.

- 2 entrées (tailles de la matrice $m \times n$, avec $m, n \in [1000, 5000]$)
- 8 paramètres de design (nombre de threads, taille de blocs, seuil de changement de stratégie, etc.) [1]

Total de 4.6×10^{13} combinaisons de design.

1. Liste exacte confidentielle.

Geometric Mean Speedup

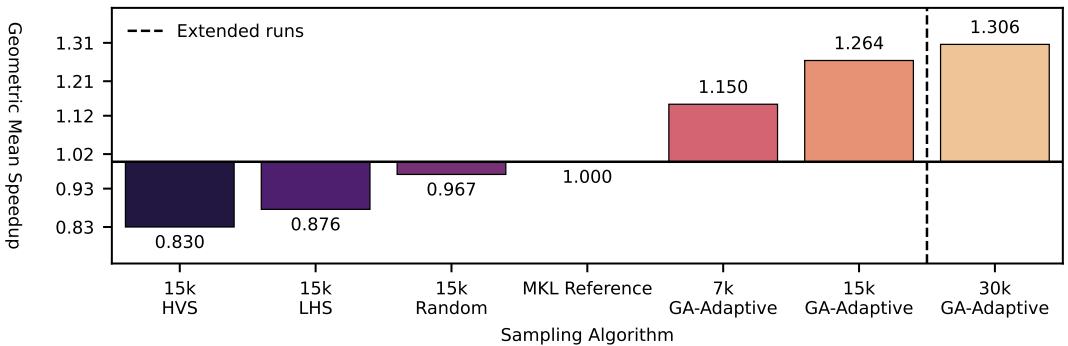


Figure – (SPR) Geomean Speedup (Higher is better) relativement aux configurations de références de la MKL sur `dgetrf` (LU) selon l'algorithme échantillonnage. Grille de validation de 46x46 sur I.

Speedup sur l'espace

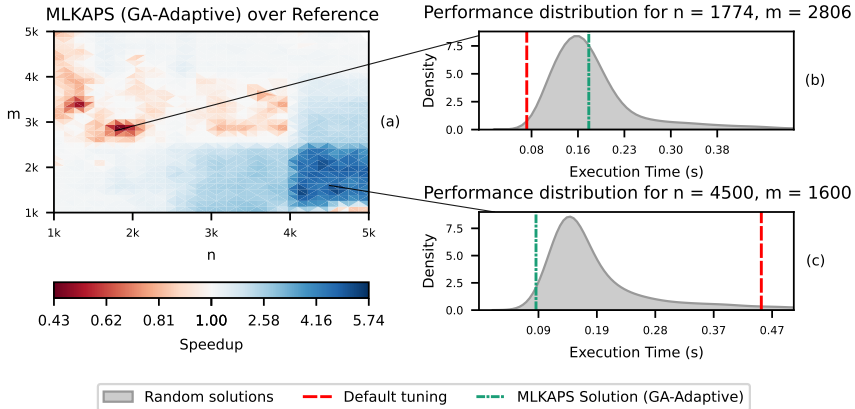


Figure – (KNM) `dgetrf` (LU) a) Speedup de GA-Adaptive (7k samples) comparer a la MKL. b/c) Distribution de la performance des configurations local.

Comparaison a l'etat de l'art

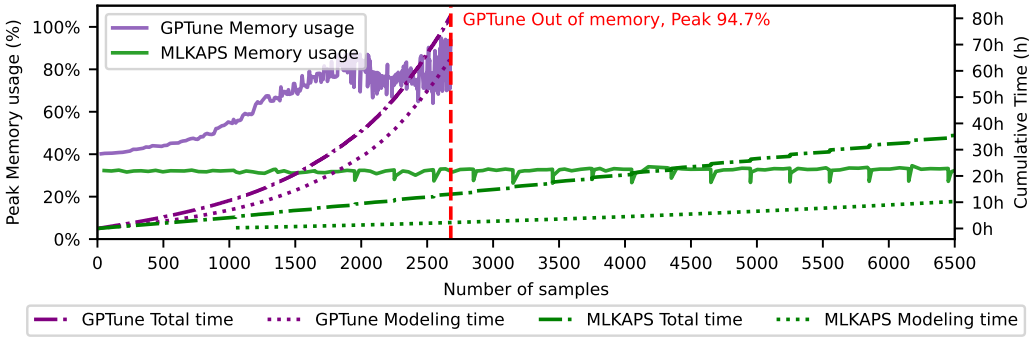


Figure – (KNM) `dgetrf` (LU) Évolution de la consommation mémoire et du temps d'exécution selon le nombre d'échantillons pour MLKAPS et GPTune (16 tasks).

1. GPTune : multitask learning for autotuning exascale applications, Yang Liu, Wissam M Sid-Lakhdar, Osni Marques, Xinran Zhu, Chang Meng, James W Demmel, and Xiaoye S Li.

Conclusions

Soumission ACM Taco under review, disponible sur HAL [2] et arXiv [3]

- Projet MLKAPS débuté par Intel en 2022, maintenant dans le domaine public `https://github.com/MLCGO/MLKAPS`
- Les arbres de décisions de MLKAPS pour QR et LU sont dans la Intel MKL depuis la release 2025.2
- Plus de resultats dans le papier.

Sinon : `mathysjam.org`

2. https ://hal.science/hal-04851397v1
3. https ://arxiv.org/abs/2501.05811