HPC TOOL PROJECT : kerncraft

Auteur : Alexandre Gabrielli

Introduction

L'outils kerncraft a pour d'effectuer une analyse statique d'un code noyau afin de donner un aperçu des goulots d'étranglement du noyaux et mémoire. Il permet de produire et récupérer les donner afin d'appliquer des modèles de performance.

Kerncraft propose donc des modèles d'analyse qui n'exécute pas réellement le code cible, il se base sur une description yaml de l'architecture cible afin de prédire les performances.

Installation

Lorsque l'on installe l'outils il nécessite IACA de Intel, pour ce faire un script python iaca_get est fourni malheureusement il n'est plus d'actualité. J'ai donc simplement réécrit rapidement le script en changeant les url de download des version d'IACA nécessaire (on peut voir dans la liste de commit git qu'ils sont en train d'essayer de fixer cella mais leur pipeline ne passe pas encore les tests).

Modèle donnée utilisé par Kerncraft

Kerncraft propose trois modèles d'analyse et de prédiction différente : ECM, Roofline et Benchmark. ECM et Roofline n'exécute pas le code à proprement parler tandis que Benchmark oui.

Roofline

Roofline est un modèle qui considéré l'intensité opérationnelle qui représente le nombre d'opérations par octet du trafic mémoire. Ce modèle permet de prédire dans un système donné ce chiffre grâce à des formule. Plus de détail dans la doc officiel.

Exécution-Cache-Memory Model (ECM)

Basé sur Roofline se modèle donne plus d'importance à la hiérarchie du cache et introduit d'autre mesure comme les opérations ponctuelles par seconde (FLOP/s) et le cycles par ligne de cache (cy/CL). (plus petite quantité de données transférable entre la mémoire principal et les niveaux de cache).

CLI

Kerncraft s'utilise principalement en ligne de commande

kerncraft --machine MACHINE --pmodel PMODEL [-D KEY VALUE] [--verbose] [--asm-block BLOCK] KERNELFILE [KERNELFILE ...]

Ou machine est le chemin vers le yaml de description de la machine

KERNELFILE est le chemin vers le code noyau que l'on veut analyser

-D permet de passer les paramètres nécessaires au code

PMODEL permet de passer le modèle de performance avec lequel on souhaite analyser le code il en existe plusieurs :

ECM : combinaison d'ECMData et d'ECMCPU

```
T_ECM = max(T_comp, sum(T_RegL1, T_L1L2, T_L2L3, T_L3MEM)) cy/CL
      = max(10.5, sum(8.0, 10.0, 10.0, 13.1)) cy/CL
      = 41.06 cy/CL
saturating at 5 cores
prediction for 1 cores, assuming static scheduling: 41.06 cy/CL (memory-interface not saturated, in-NUMA-domain scaling)

Scaling prediction, considering memory bus utilization penalty and assuming all scalable caches:
cores         || 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8
perf. (cy/CL) || 41.1  | 21.6  | 16.3  | 14.2  | 13.1  | 13.1  | 13.1  | 13.1
```

cette outils nous permet de savoir a combien de cores théoriquement nous serons saturé. Et nous donne pour chaque nombre de cores une approximation du cy/CL(cycles par ligne de cache = plus petite quantité de données transférable entre la mémoire principal et les niveaux de cache). Cette donnée est très interessante car elle nous permet de savoir (sans avoir les CPU physiquement puisque tout ce fait en static) lorsque l'on cherche la scalabilité a combien de cores il faut théoriquement s'arrêter.

ECMData

```
(base) alexandre@alexandre-Mint:~/Bureau/kerncraftTest$ kerncraft -p ECMData -m Zen_Ryzen7-1700X.yml 2d-5pt.c -D N 10000 -D M 10000
                                kerncraft
2d-5pt.c                                                -m Zen_Ryzen7-1700X.yml
-D N 10000 -D M 10000
---------------------------------- ECMData ----------------------------------
L2 = 8.00 cy/CL
L3 = 4.00 cy/CL
MEM = 12.19 cy/CL
```

ECMData nous donne toujours les cy/CL mais ici en donnant par niveau de cache et memoire système.

EMCPU : ECM d'Intel (via IACA)

```
(base) alexandre@alexandre-Mint:~/Bureau/kerncraftTest$ kerncraft -p ECMCPU -m SkylakeSP_Gold-6148_SNC.yml 2d-5pt.c -D N 10000 -D M 10000
                                kerncraft
2d-5pt.c                                -m SkylakeSP_Gold-6148_SNC.yml
-D N 10000 -D M 10000
---------------------------------- ECMCPU -----------------------------------
In file included from .2d-5pt.c_kerncraft/kernel.c:1:0:
../../anaconda3/lib/python3.7/site-packages/kerncraft/headers/kerncraft.h: In function 'aligned_malloc':
../../anaconda3/lib/python3.7/site-packages/kerncraft/headers/kerncraft.h:17:13: warning: implicit declaration of function 'posix_memalign'; did you mean '__builtin_posi
x_memalign'? [-Wimplicit-function-declaration]
         if(posix_memalign(&result, align, size)) result = 0;
            ^~~~~~~~~~~~~~
            __builtin_posix_memalign
T_comp = 3.3 cy/CL
T_RegL1 = 2.0 cy/CL
```

il faut s'avoir que cette commande ne marche que sur les architectures intel (normal puisque IACA est fait pour et par intel), elle nous donne un T_COMP qui est une moyenne des niveau de cache supérieur a L1 et aussi un T_RegL1 qui est un resultat pour la cache L1

Roofline : Roofline "custom" de Kerncraft

```
(base) alexandre@alexandre-Mint:~/Bureau/kerncraftTest$ kerncraft -p Roofline -m SkylakeSP_Gold-6148_SNC.yml 2d-5pt.c -D N 10000 -D M 10000
                                kerncraft
2d-5pt.c                                -m SkylakeSP_Gold-6148_SNC.yml
-D N 10000 -D M 10000
---------------------------------- Roofline ---------------------------------
Cache or mem bound.
5.98 GFLOP/s due to L3 transfer bottleneck (with bw from update benchmark)
Arithmetic Intensity: 0.12 FLOP/B
```

Cette outils nous permet de detecter les goulets d'étranglement au niveau de la mémoire, on voit ici que le goulet d'étranglement ce situe au niveau de la cache L3 (FLOP/s = opérations ponctuelles par seconde) et nous donne l'intensité arithmetic en FLOP / Bytes.

RooflineIACA : donne le résultat d'un Roofline Intel avec iaca

```
(base) alexandre@alexandre-Mint:~/Bureau/kerncraftTest$ kerncraft -p RooflineIACA -m SkylakeSP_Gold-6148_SNC.yml 2d-5pt.c -D N 10000 -D M 10000
                                kerncraft
2d-5pt.c                                -m SkylakeSP_Gold-6148_SNC.yml
-D N 10000 -D M 10000
------------------------------ RooflineIACA -------------------------------
Cache or mem bound.
5.98 GFLOP/s due to L3 transfer bottleneck (with bw from update benchmark)
Arithmetic Intensity: 0.12 FLOP/B
```

même chose que précedament mais cette commande ne marche que pour les architectures intel.

Benchmark

Nous n'avons pas pu tester cette fonctionnalité a cause d'un BUG (voir section suivante)

bug

Normalement ont généré la description de notre machine en utilisant likwid_bench_auto et ont rempli les champs manquant (REQUIRED_INFORMATION) mais bien sur ce script ne fonctionne pas. Essayer de débugger deviens très compliquer car on peut qualifier les scripts de "code spaghetti" tellement il est complexe de le lire. Nous allons donc prendre un yaml de description d'architecture et utilisé pour analyser avec les deux modèle ECM et Roofline qui n'exécute pas réellement le code. ici nous nous baserons uniquement sur des yaml fourni.b

Conclusion

Kerncraft est un outil extrêmement intéressant combinant deux modèles d'HPC. Il permet grâce à sa description yaml de pouvoir prédire des résultats via une définition de la machine sans devoir possèder les CPUs. On peu donc l'utilisé pour le choix d'un processeur lorsque l'on a une opération spécifique a réalisé. On peu aussi grace au modèle ECM utilisé par kerncraft savoir combien de cores on doit acheter pour une scalabilité maximal. On voit aisément l'utilité de cette outils si on désire construire un système répartie (combien de cores sur chaque machine en prennant en compte en plus le transfert de donné entre machine).

Malheureusement le code python (principal langage de Kerncraft) est un code spaghetti est on voit, lorsqu'on regarde les commit du repo git ,que son créateur a énormément de mal à le maintenir. Lorsque l'on veut l'utiliser on se retrouve souvent à devoir débugger des bouts de code ce qui rend son utilisation extrêmement complexe.

De plus la documentation est inéxistante, la seul "doc" est la thése de master qui date de 2015 et qui n'est donc plus à jour.

Kerncraft est un très bonne outils car il propose une méthode de calcul statique qu'on retrouve peu mais souffre d'un manque de rigueur au niveau de son code (kerncraft n'était pas le but premier de la thèse de master mais bien les calculs qui sont dessous) qui rend son maintient quasiment impossible.