HPC - Laboratoire 6

Consommation energétique

Rayane Annen

28 mai 2024

Table des matières

Machine de test

Machine utilisée pour les tests :

- Architecture : Intel x86_64
- CPU : Intel Core i7-1165G7 @ 2.8 GHz / Turbo @ 4.7 GHz
  - Cache L1d : 192 KiB
  - Cache L1i : 128 KiB
  - Cache L2 : 5 MiB
  - Cache L3 : 12 MiB
- OS: Ubuntu 24.04
- Compilateur :
  - gcc 13.2.0
  - target: x86_64-linux-gnu
  - Flags de compilation: `-O3 -g -Wall -fno-inline -march=native -fno-tree-vectorize`
  - Librairies: `stb`, `math.h`, `Likwid`

Outils :

- Perf
- Powercap : code fourni mais pas possible de run le programme

N.B : Pas possible d'utiliser likwid car incompatible avec ma machine.

Capture des performances

Paramètres utilisée sur le programme de segmentation d'imageö

- image `nyc.png` (c.f. précédents laboratoires pour avoir plus d'informations sur l'image)
- Nombre de clusters 10

`perf`

Consommation d'énergie [J]	Evènement	Programme
6.49	power/energy-cores/	segmentation
6.46	power/energy-cores/	segmentation_simd
0.06	power/energy-gpu/	segmentation
0.04	power/energy-gpu/	segmentation_simd
7.74	power/energy-pkg/	segmentation
7,67	power/energy-pkg/	segmentation_simd
2.17	power/energy-psys/	segmentation
2.17	power/energy-psys/	segmentation_simd

Mesures plus précises en utilisant les marqueurs directement, j'ai décidé de mesurer la fonction `kmeans_pp` :

```
void kmeans(struct img_1D_t *image, int num_clusters){
    uint8_t *centers = calloc(num_clusters * image->components, sizeof(uint8_t));
    perf_manager pmon;
    perf_manager_init(&pmon);


    perf_manager_resume(&pmon);
    // Initialize the cluster centers using the k-means++ algorithm.
    kmeans_pp(image, num_clusters, centers);
    perf_manager_pause(&pmon);
    // ....
}
```

Consommation d'énergie [J]	Evènement	Programme
1.73	power/energy-cores/	segmentation
1.88	power/energy-cores/	segmentation_simd
0.01	power/energy-gpu/	segmentation
0.01	power/energy-gpu/	segmentation_simd
2.27	power/energy-pkg/	segmentation
2.17	power/energy-pkg/	segmentation_simd
0.60	power/energy-psys/	segmentation
0.80	power/energy-psys/	segmentation_simd

Analyse des résultats

D'un point de vue consommation, la version SIMD et la version normale semble être sensiblement pareille.

En effet, on remarque même que lorsqu'on compare directement un appel de fonction SIMD et un autre qui ne l'est pas la version SIMD consomme un peu plus d'énergie, ce qui peut s'expliquer par le fait que SIMD car ce sont des opérations qui sont plus intensives.

Toutefois, lorsqu'on compare l'énergie consommée sur une opération qui prend plus de temps (N cluster = 100), la différence se voit beaucoup plus :

Consommation d'énergie [J]	Evènement	Programme
50.84	power/energy-cores/	segmentation
21.82	power/energy-cores/	segmentation_simd
0.03	power/energy-gpu/	segmentation
0.11	power/energy-gpu/	segmentation_simd
58.08	power/energy-pkg/	segmentation
24.79	power/energy-pkg/	segmentation_simd
24.60	power/energy-psys/	segmentation
7.08	power/energy-psys/	segmentation_simd

Consommation d'énergie [J]	Evènement	Programme

Comme on le voit, le gain de performances des instructions SIMD joue beaucoup, en effet, comme le temps de calcul est plus faible (env. 2x plus rapide pour SIMD comparée à la version normale), la consommation est forcément moindre.

Pour résumer :

- Sur des petits calculs la consommation est pareille voir à peine plus pour SIMD.
- Sur des calculs plus longs : les instructions SIMD consomment clairement moins (car moins de temps de calcul intensif).

D'un point de vue zone de consommation, on constate que la consommation d'énergie du GPU est négligeable car on ne l'utilise pas.

Concernant la zone plateforme (psys), je m'attendais à voir une consommation plus élevée que le reste (puisque c'est la zone qui représente la consommation totale), mais je ne suis pas capable de l'expliquer.

`powercap`

J'ai voulu réaliser cette partie mais je me suis rendu compte après avoir programmé que je ne pouvais pas lancer le programme (erreur d'initialisation), même en étant un utilisateur privilégié (sudo). J'ai essayé d'installer la librairie depuis le repo github mais sans succès également.

Malheureusement je ne suis pas en capacité d'effectuer les mesures, par conséquent je ne peux que fournir le code afin de lancer les mesures.

Code permettant de mesurer :

```
void kmeans(struct img_1D_t *image, int num_clusters) {
  // ...
    uint32_t npackages = powercap_rapl_get_num_instances();
    powercap_rapl_pkg pkg[npackages];
    uint64_t energy_uj1[npackages];
    uint64_t energy_uj2[npackages];

    powercap_rapl_zone zone = POWERCAP_RAPL_ZONE_CORE;

    for (size_t i = 0; i < npackages; ++i) {
        if (powercap_rapl_init(i, &pkg[i], true)) {
            printf("Error initializing RAPL, you may need privileged access\n");
            powercap_rapl_destroy(&pkg[i]);
            return;
        }
    }

    bool supported[npackages];
    bool has_one_supported = false;

    for (size_t i = 0; i < npackages; ++i) {
        supported[i] = powercap_rapl_is_zone_supported(&pkg[i], zone);
        if (supported[i] != 1) {
            printf("RAPL is not supported on packages %ld\n", i);
        } else {
            has_one_supported = true;
        }
    }

    if (!has_one_supported) {
```

```
        printf("No supported package\n");
        return;
    }

    for (size_t i = 0; i < npackages; ++i) {
        if (supported[i]) {
            if (powercap_rapl_get_energy_uj(&pkg[i], zone, &energy_uj1[i])) {
                printf("Failed to get energy on package %ld\n", i);
                break;
            }
        }
    }


    // Initialize the cluster centers using the k-means++ algorithm.
    kmeans_pp(image, num_clusters, centers);

    for (size_t i = 0; i < npackages; ++i) {
        if (supported[i]) {
            if (powercap_rapl_get_energy_uj(&pkg[i], zone, &energy_uj2[i])) {
                printf("Failed to get energy on package %ld\n", i);
                break;
            }
        }
    }
```