Système d'exploitation

L2 Informatique – UVSQ

Sebastien GOUGEAUD

pro.seb.gougeaud@gmail.com

Fenêtre sur le HPC

Qu’est-ce que le HPC ?

Calcul Haute Performance – *High Performance Computing*

→ Branche de l’informatique traitant des supercalculateurs

- Utilisés pour la modélisation et la simulation de problèmes
- Problèmes qui ne cessent de croître en :
  - temps de calcul
  - taille de données.

Quelques chiffres

TOP500 : classement bi-annuel des supercalculateurs

Première place détenue par Fugaku (*RIKEN Center for Computational Science*, Japan) :

- Nombre de coeurs : 7’630’848
- Performance max : 442’010 TFLOPS
- Puissance énergétique : 29’899 kW

Green500 : classement fait sur le rapport performance/puissance énergétique

Centre de calcul

Un centre de calcul est constitué de plusieurs noeuds de calcul organisés en îlot : les noeuds placés sur le même îlot ont des connexions plus directes qu'avec les autres

Peut être constitué de plusieurs technologies et/ou processeurs

Centre de données

Un centre de calcul est généralement accompagné d'un centre de données, pour y stocker les données des programmes

- Systèmes se composant de Petaoctets, voire Exaoctets de données
- Généralement constitué de plusieurs technologies (Flash, rotatif, bandes magnétiques, ...)

Quel est le rapport avec le système d'exploitation ?

- Il faut toujours pouvoir assurer la liaison entre les applications et les ressources matérielles
- Il faut toujours pouvoir assurer le partage *équitable* des ressources entre les applications

- Utilisation d'allocateur de tâches (slurm par exemple)
- Appels systèmes effectués par des bibliothèques utilisées dans le milieu du HPC

Calcul parallèle : OpenMP

- Utilisation en mémoire partagée
- Apport d'une API *simple* pour le parallélisme des threads

Calcul parallèle : OpenMP

```c
int main(void) {
    int local_sum = 0, global_sum = 0;
    int TAB[16384];

    #pragma omp parallel private(local_sum) shared(global_sum)
    {
        #pragma omp for
        {
            for (int i = 0; i < 16384; ++i)
                local_sum += TAB[i];
        }

        #pragma omp critical
        {
            global_sum += local_sum;
        }
    }

    printf("Global sum: %d\n", global_sum);

    return 0;
}
```

Calcul parallèle : MPI

- Utilisation en mémoire distribuée
- Chaque processus MPI possède un identifiant
- Communications : pair à pair, diffusion, groupes

Calcul parallèle : MPI

```c
int main(void) {
    int id;

    MPI_Init(NULL, NULL);
    MPI_Comm_rank(MPI_COMM_WORLD, &id);

    if (id == 0) {
        printf("Hello, I'm process #%d, and you are?\n", rank);
        MPI_Send(&id, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
    } else {
        int msg;

        MPI_Recv(&msg, 1, MPI_INT, MPI_ANY_SOURCE,
                 MPI_ANY_TAG, MPI_COMM_WORLD);
        printf("Hi, I'm #%d, nice to meet you %d!\n", rank, msg);
    }

    MPI_Finalize();

    return 0;
}
```

Calcul parallèle : CUDA

- Utilisation sur processeur graphique
- Très efficace en SIMD (*Single Instruction Multiple Data*)
- Coût des communications entre RAM et GPU non négligeables

Calcul parallèle : CUDA

```
1  int main(void) {
2      float *a, *b, *out;
3      float *d_a;
4
5      a = (float*)malloc(sizeof(float) * N);
6      cudaMalloc((void**)&d_a, sizeof(float) * N);
7
8      cudaMemcpy(d_a, a, sizeof(float) * N, cudaMemcpyHostToDevice);
9
10     ...
11     vector_add<<<1,1>>>(out, d_a, b, N);
12     ...
13
14     cudaFree(d_a);
15     free(a);
16
17     return 0;
18 }
```

Entrées/sorties parallèles

- MPI-IO : s'appuie sur MPI pour l'écriture de fichiers de large taille en parallèle
- HDF5 : format de fichier de large taille organisé en conteneurs de données