Modèles et techniques en programmation parallèle hybride et multicœurs

Programmation hybride MPI – Cuda ou OpenCL

Marc Tajchman

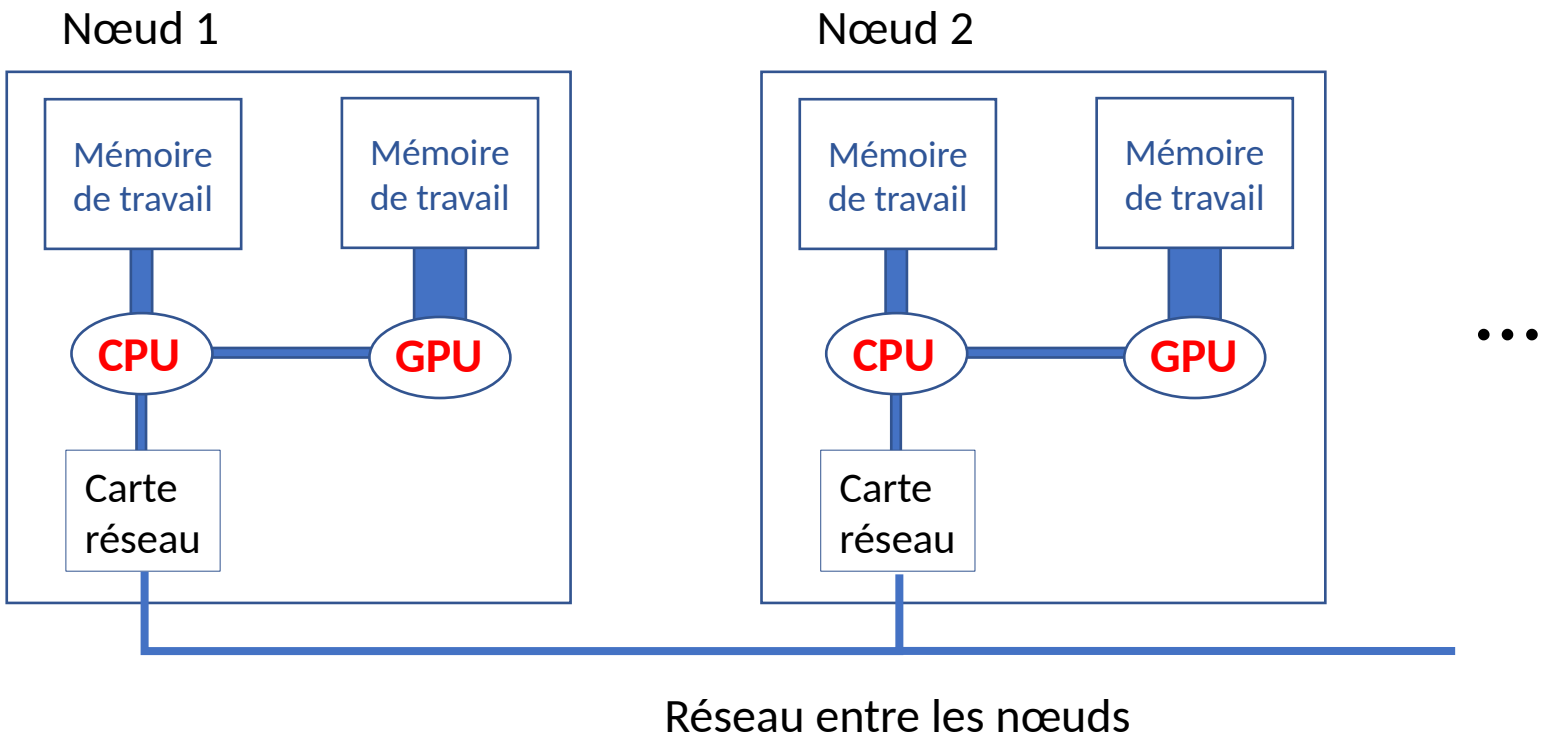
CEA - DEN/DM2S/STMF/LMES

08/02/2022

Environnement matériel:

On considère une machine parallèle avec n nœuds, chacun contenant un (ou plusieurs) CPU et un (ou plusieurs) GPU.

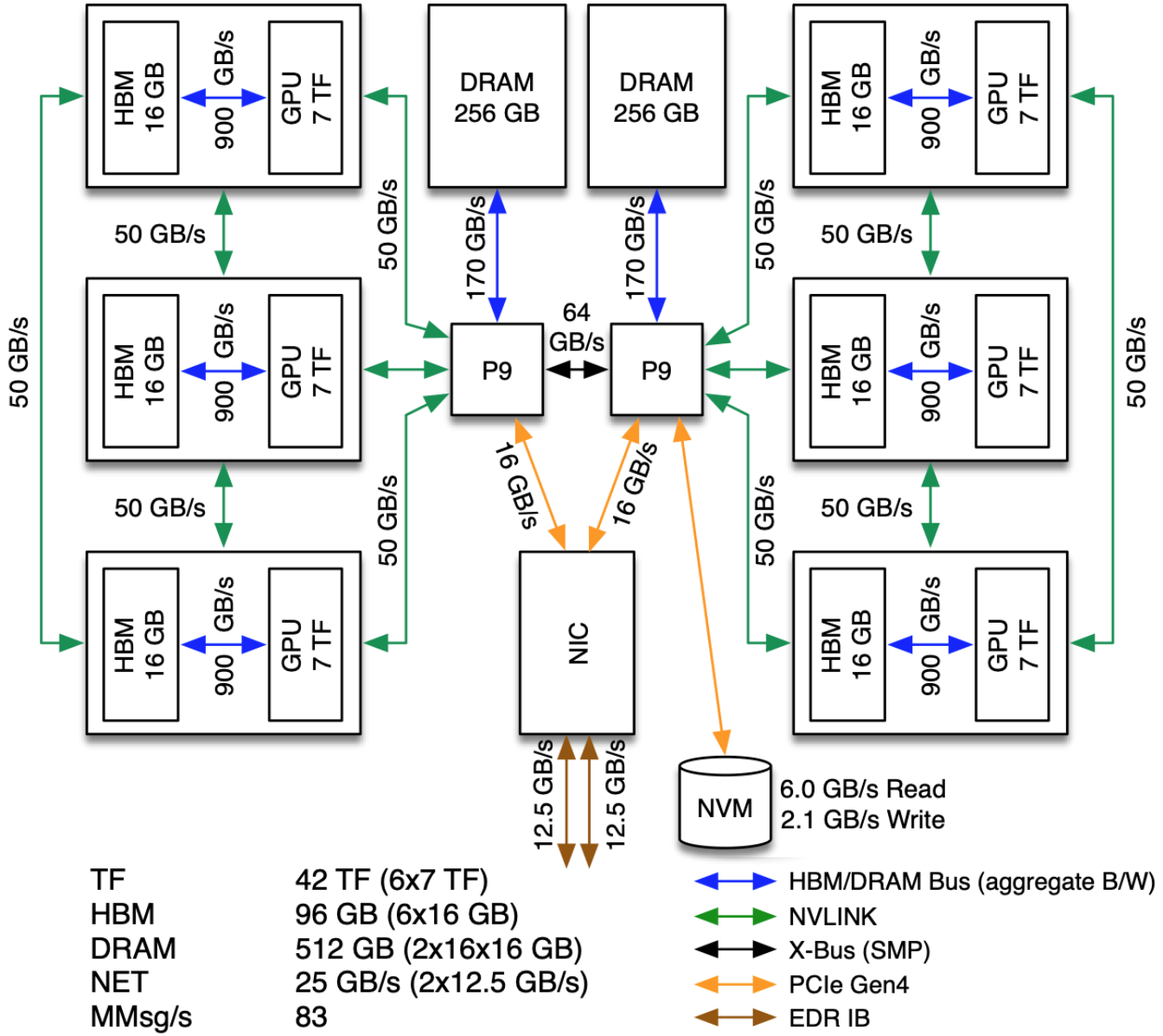
Dans chaque nœud, le(s) CPU et le(s) GPU sont associés à leur mémoire de travail propre.



Exemple : machine Summit (Oak Rigde Nat. Lab, USA)

4806 nœuds chacun avec 2 CPU (IBM Power PC) + 6 GPU(Nvidia Volta)

Schéma de l'intérieur d'un nœud:

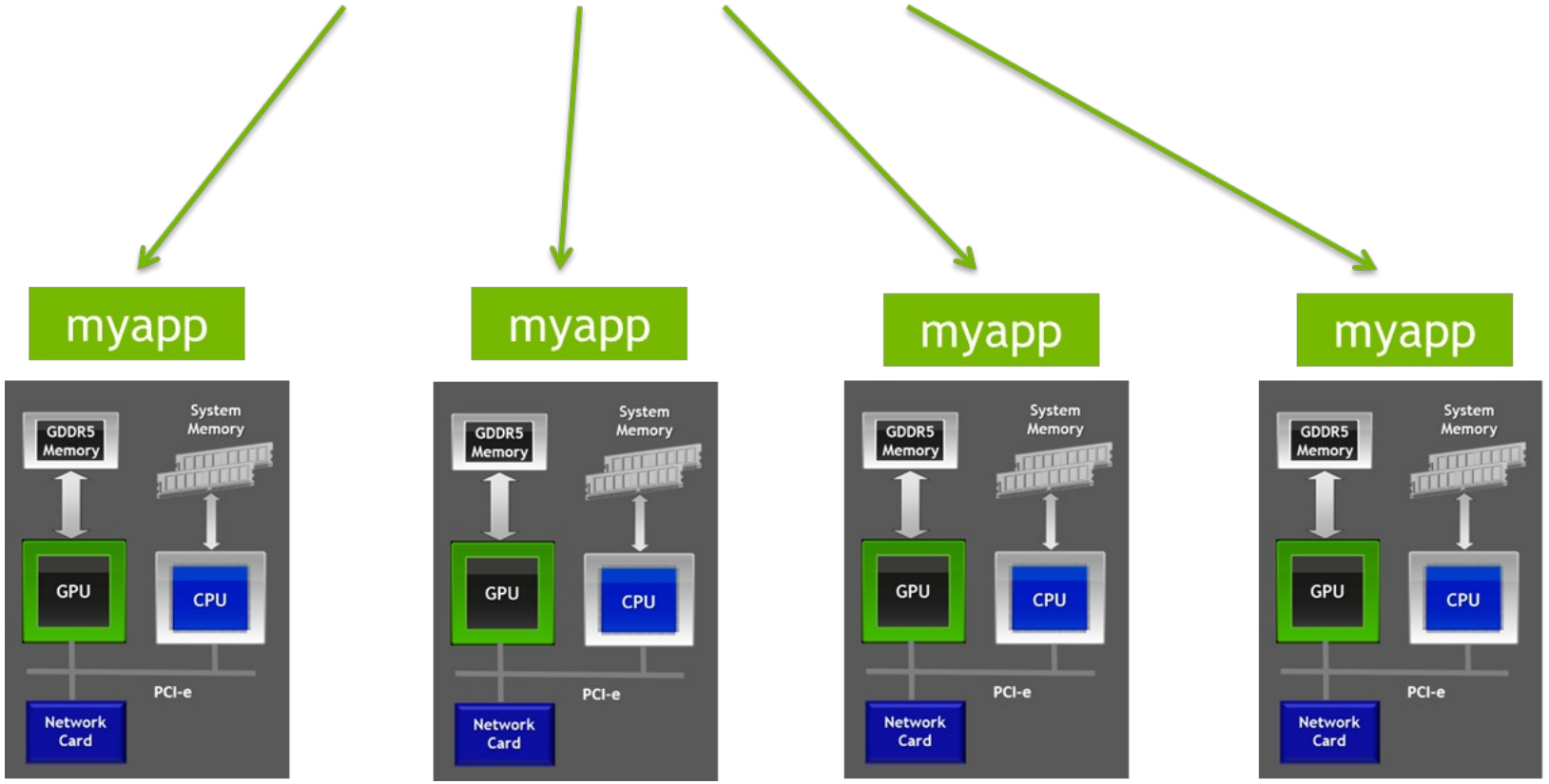


Sur ce type de machine, on a la possibilité de

- Créer un (ou plusieurs) processus MPI dans les nœuds utilisés
- Dans chaque processus MPI, créer plusieurs threads (si le processus MPI contrôle plusieurs cœurs de CPU)
- Lancer des calculs sur la (ou les cartes) graphiques contrôlées par le processus MPI

Dans cet exposé, on supposera qu'on lance un processus MPI par nœud utilisé et que chaque nœud contient un GPU

```
mpirun -np 4 ./myapp <args>
```



Source : https://developer.nvidia.com/blog/introduction-cuda-aware-mpi

La programmation combinée MPI - Cuda (ou OpenCL) n'est pas compliquée si on garde à l'esprit quelques règles

- C'est le CPU qui fait les appels MPI (pas d'appels MPI dans les noyaux Cuda ou OpenCL)

  MPI permet d'échanger des données entre les mémoires de travail des CPU

- Si on a besoin d'envoyer une donnée de la mémoire d'un GPU à un autre GPU sur un autre nœud, il faut:

  1. Dans le nœud de départ, copier la donnée de la mémoire du GPU sur la mémoire du CPU sur le même nœud
  2. Envoyer la donnée sur la mémoire du CPU du nœud d'arrivée
  3. Dans le nœud d'arrivée, copier la donnée de la mémoire du CPU vers la mémoire du GPU sur le même nœud

- Faire attention aux barrières : dans MPI (MPI_Barrier), dans CUDA (cudaDeviceSynchronize)
- Même si ce n'est pas obligatoire, mettre les appels MPI dans les fichiers compilés par le compilateur C/C++ (gcc/g++ ...) et pas dans les fichiers compilés par le compilateur Cuda (nvcc)

Dans les versions récentes de MPI (dites « MPI-Cuda aware ») il existe certaines simplifications/optimisations de ces opérations.

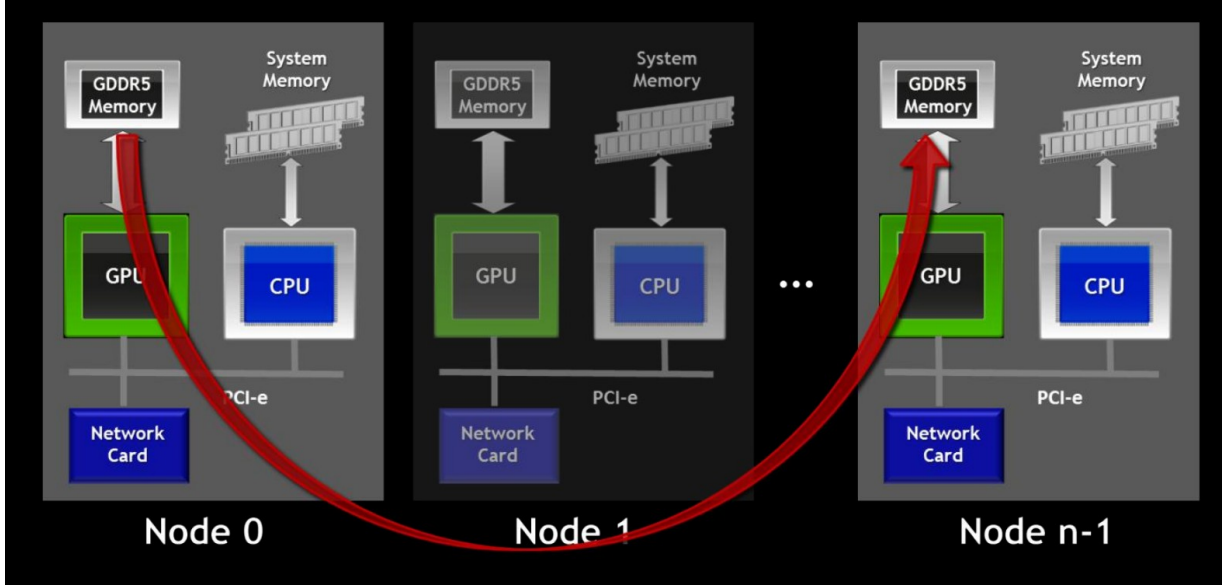
Par exemple, on peut fournir à MPI_Send/MPI_Recv, des pointeurs vers la mémoire du GPU. C'est MPI_Send qui copiera les données de la mémoire du GPU vers le réseau et inversement pour MPI_Recv.

Mais cela demande des versions de MPI spécialement compilées (on vérifiera si c'est la cas pour les versions de MPI disponibles dans les machines de calcul visées) et des GPU compatibles.

Exemple de programmation hybride MPI – Cuda : Exemple10.tgz

Exemple:

Dans chaque noeud :
- A_d est un vecteur dans la mémoire d'un GPU
- A_h est la copie de A_d dans la mémoire du CPU associé



Cuda + MPI (non Cuda – aware)

```
//MPI rank 0
cudaMemcpy(A_h, A_d, size, …);
MPI_Send(A_h, size, …);
```

```
//MPI rank n-1
MPI_Recv(A_h, size, …);
cudaMemcpy(A_d, A_h, size, …);
```

Cuda + MPI (Cuda – aware)

```
// MPI rank 0
MPI_Send(A_d,size,…);
```

```
//MPI rank n-1
MPI_Recv(A_d,size,…);
```