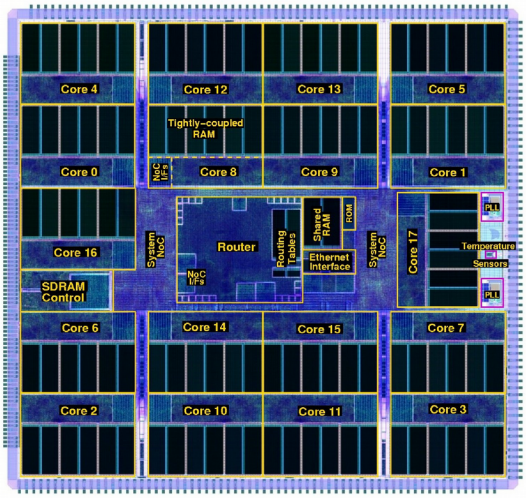
SpiNNaker

Le Neuromorphic Computing Platform du HBP contient deux calculateurs neuromorphiques (dont l'architecture se rapproche de celle du cerveau) : “BrainScaleS” (à Heidelberg, « physical model ») et “SpiNNaker” (à Manchester, « many core »). Nous allons ici présenter le calculateur SpiNNaker (Universal Spiking Neural Network Architecture).

1 - Architecture physique

La machine SpiNNaker est basée sur une architecture multi-cœurs dans le but de simuler un milliard de neurones en temps réel.

En mars 2016, SpiNNaker comportait 28 800 puces, ce qui correspond à 518 400 cœurs.



Une puce (aussi appelée nœud) comporte 18 cœurs ARM9 encerclant un routeur, une mémoire partagée et des tables de routages ainsi que 128 Mo de mémoire SDRAM.
Une seule puce, d’une surface de 10 mm sur 9,7 mm, peut simuler 16 000 neurones avec 8 millions de synapses. Le système simule actuellement près de un demi-milliard de neurones.

Chaque puce a 6 liaisons bidirectionnelles inter-puce qui permettent de former des réseaux de différentes topologies.
Chaque cœur ARM9 (choisi pour sa faible consommation d’énergie) correspond à un processeur avec 64 Ko de mémoire de données fortement couplée et 32 Ko de mémoire d’instruction fortement couplée.

Le SpiNNaker utilisé par le HBP dans sa version actuelle est équivalent aux cerveaux de plusieurs souris. Il existe plusieurs machines de type SpiNNaker : la Jorg Conradt’s avec une carte à une seule puce (pour les applications robotiques nécessitant un poids léger), la 4-nodes board (4 puces), la 48-nodes board (48 puces). Ici, dans le cadre du HBP, la machine SpiNNaker a 600 cartes de 48 puces chacune, chaque carte pouvant être utilisée seule. Il est donc possible d’effectuer aussi bien une seule grande simulation mobilisant toutes les cartes que plusieurs petites simulations isolées et simultanées.

2 – Fonctionnement

À l'intérieur d'une puce, la communication se fait par multicast (diffusion à plusieurs destinataires), avec commutation de paquets de 40 bits (5 octets) ou de 72 bits (9 octets). La bande passante peut aller jusqu'à 5 milliards de paquets/seconde. Chaque paquet ne porte que des informations sur l'émetteur et le routeur est chargé de la distribution vers le(s) destinataire(s). On a ainsi un nombre très important de paquets de très petite taille en circulation, appelés « spikes ».

Ce système possède cependant une limite, car même si la bande passante est très importante, des « goulots » (bottleneck) se forment lors de l'accès en écriture à la mémoire.

Étant donné l'énorme taille du système et le nombre très important de composants, il est inévitable que certains rencontrent des problèmes de panne et/ou de transmission des données. Pour gérer cela, une détection d'erreurs et une récupération des données sont intégrées au système.

Il existe une interface web d'accès unique aux deux calculateurs BrainScale et SpiNNaker : la HBP Neuromorphic Computing Platform. Celle-ci donne la possibilité de soumettre des jobs. Il suffit de rejoindre la communauté HBP en demandant un compte et de spécifier les ressources dont on a besoin. Cela va créer un espace collaboratif mettant à disposition les outils nécessaires pour l'accès à la plateforme. On peut alors ajouter des membres à l'espace et exécuter des simulations. Il y a 3 types d'accès :

- accès « Test » : seul un résumé de l'utilisation que l'on souhaite faire de la plateforme est requis. Un quota de 5000 coeurs-heure est alloué ainsi qu'un espace de stockage temporaire de 1 Go.
- accès « Préparatoire » : un document expliquant l'utilisation souhaitée ainsi qu'une première expérience sur la plateforme sont requis.
- accès en mode « Projet » : pour des quotas plus importants, une page décrivant la motivation scientifique du projet ainsi qu'une demande des ressources nécessaires doivent être fournies.

Quand on utilise le portail web HBP pour soumettre un job, SpiNNaker partitionne automatiquement la population définie dans le script sur les cœurs, de façon optimale selon les ressources disponibles. Le processus de mapping examine la définition du réseaux neuronal et le partitionne pour le distribuer sur les différents cœurs. Il est possible d'influencer le partitionnement en ajoutant par exemple des contraintes pour limiter le nombre de neurones sur chaque cœur. L'algorithme de routage est ensuite exécuté pour calculer les routes empruntées par les paquets entre les cœurs.

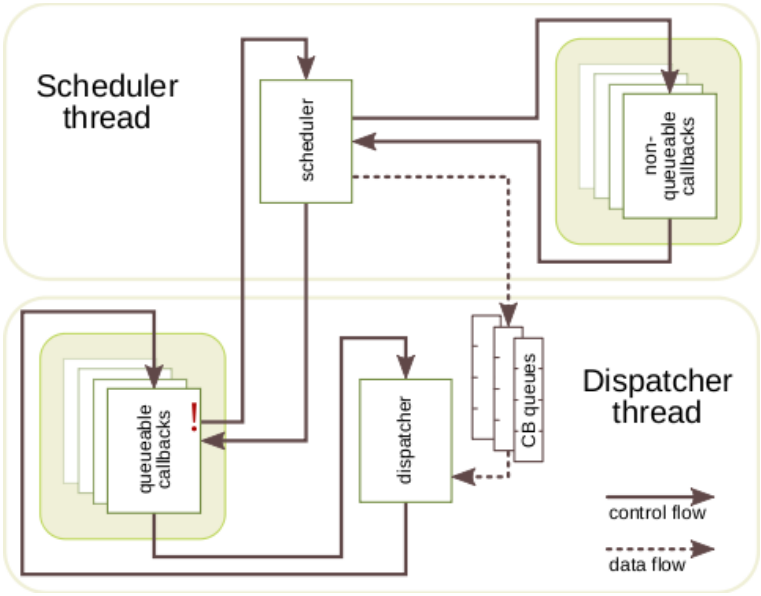
Au démarrage, un des cœurs est élu au rôle de « cœur moniteur » et gère le système. 16 cœurs sont dédiés à l'exécution de l'applicatif et un cœur est réservé pour la tolérance aux pannes et l'amélioration du rendement de fabrication.

Les principaux axiomes spécifiques au calcul haute performance ne sont pas respectés : cohérence mémoire, synchronisation et déterminisme. Au lieu de ça, de simples messages (spikes) sont routés dynamiquement.

Le système SpiNNaker repose sur plusieurs principes.

- La puce est Globally Asynchronous, Locally Synchronous : chaque bloc la composant est synchrone, utilise sa propre horloge et communique avec les autres blocs de façon asynchrone.
- Event driven model : il n'y a pas de contrôle de l'exécution des fonctions mais des appels à base de callbacks. On associe des appels de fonctions à des événements et à une priorité.

Quand un événement (arrivée d'un paquet par exemple) survient, la fonction correspondante est exécutée. Selon le type de callback (queueable ou non queueable), soit le scheduler exécute la fonction immédiatement et atomiquement, soit il la place dans une queue selon sa priorité. Quand le callback en cours se termine, le suivant dans la queue est exécuté. L'exécution des callback en queue n'est pas forcément atomique. En effet, si un événement non queueable arrive pendant l'exécution d'un callback queueable, il est possible que celui-ci soit interrompu. Si la queue est vide, le dispatcher dort et se réveille quand un événement arrive.



- La simulation en temps réel. Le but final est de simuler le fonctionnement d'un cerveau en temps réel. Pour simuler un milliard de neurones, il faut 50 000 puces. Cela explique le fait que l'on autorise la perte de certains paquets, cela se rapproche de la réalité dans laquelle certains spikes ne peuvent pas toujours être transmis dans le cerveau.

3 - En pratique

Modéliser des réseaux neuronaux est quelque chose de long et coûteux, le matériel est difficile à utiliser pour des non experts. C'est pourquoi, l'API PyNN est utilisée pour écrire les scripts Python composant les jobs à exécuter. Un job correspond à un script (desciption experiment), des données en entrée et une configuration du matériel (module pyNN.spiNNaker). PyNN est une API Python permettant de définir des modèles de réseaux neuronaux indépendants du simulateur utilisé. Cela permet d'avoir un haut niveau d'abstraction et de se concentrer sur les populations de neurones et les connexions en gardant accès aux neurones et synapses de façon individuelle. PyNN fournit des

modèles de neurones, synapses et connexions standards (donnant le même résultat sur différents simulateurs), qui seront ensuite traduits en noms spécifiques au simulateur effectivement utilisé. On écrit ainsi le modèle une fois et pour l'exécuter sur plusieurs simulateurs, il suffit de changer le nom du simulateur importé. Si on veut utiliser un seul simulateur, on peut utiliser les modèles de neurones et de synapses spécifiques à celui-ci (« native ») sans se restreindre aux standards.

PyNN utilise la reconfigurabilité des cœurs ARM et du routeur. L'utilisateur peut tester plusieurs topologies, plusieurs modèles de neurones et récupérer le résultat dans un format standard (NeuroTools).

Pour le simulateur SpiNNaker, l'implémentation de PyNN s'appelle spyNNaker. Cependant ce module possède certaines limitations :

- La version de PyNN utilisée n'est pas la plus récente (v0.7 au lieu de v0.8).
- Toutes les fonctionnalités ne sont pas supportées : il manque certains modèles et certaines classes ne sont pas disponibles (PopulationView, Assembly,…).
- Les modèles neuronaux ont une limitation de 255 neurones par cœur.

Pour écrire un script PyNN, il faut :

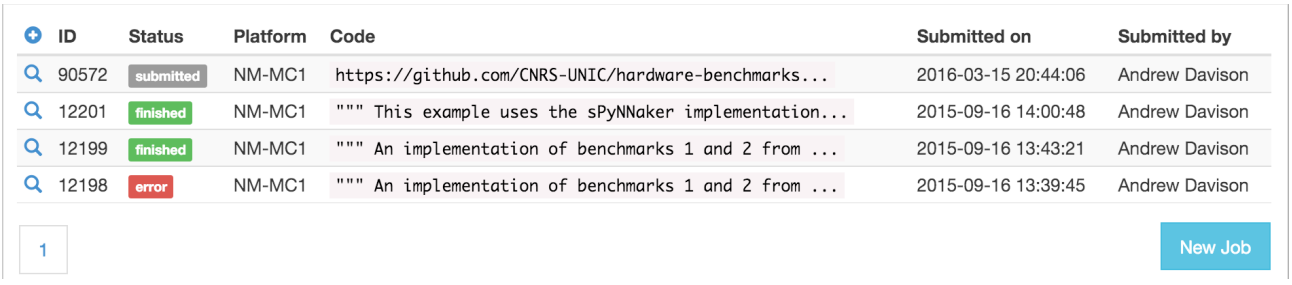
1) Importer les bibliothèques nécessaires,

2) Setup : définir la population, les propriétés des neurones, des synapses.

3) Exécuter (run) la simulation pour un temps fixe.

4) Récupérer les enregistrements dans un fichier. On peut ensuite utiliser des outils pour analyser et visualiser les résultats. Il est possible de copier les données générées vers une partie de la plateforme dédiée au stockage à long terme.

Une fois le job soumis, il apparaît dans la liste des jobs. A moins que la plateforme ne soit très utilisée, le job s'exécute en quelques minutes. Quand la simulation commence et lorsqu'elle est terminée, l'utilisateur est informé par mail et le status du job passe de « submitted » à « finished » :



ID	Status	Platform	Code	Submitted on	Submitted by
90572	submitted	NM-MC1	https://github.com/CNRS-UNIC/hardware-benchmarks...	2016-03-15 20:44:06	Andrew Davison
12201	finished	NM-MC1	""" This example uses the sPyNNaker implementation...	2015-09-16 14:00:48	Andrew Davison
12199	finished	NM-MC1	""" An implementation of benchmarks 1 and 2 from ...	2015-09-16 13:43:21	Andrew Davison
12198	error	NM-MC1	""" An implementation of benchmarks 1 and 2 from ...	2015-09-16 13:39:45	Andrew Davison

1

New Job

Sources:

https://electronicvisions.github.io/hbp-sp9-guidebook/index.html

http://spinnakermanchester.github.io/docs

http://apt.cs.manchester.ac.uk/projects/SpiNNaker

« Overview of the SpiNNaker system architecture », Steve B. Furber, David R Lester, Luis A. Plana, Jim D. Garside, Eustace Painkras, Steve Temple and Andrew D. Brown.

« Comparing Neuromorphic Solutions in Action: Implementing a Bio-Inspired Solution to a Benchmark Classification Task on Three Parallel-Computing Platforms », Alan Diamond, Thomas Nowotny, Michael Schmuker