Optimisations de performances

Hadrien Grasland

2021-12-16

Au programme

- **Upgrade RAM serveur-paon4**
- Organisation des paquets de données
- Intégration du nouveau corrélateur à Tacq
- Méthodes de travail git

Rappel de perf mémoire

- La perf de calcul est souvent limitée par la **perf mémoire**
- Les CPUs actuels ont donc plusieurs canaux mémoire
  - Perf optimale si nb barettes RAM = N x nb canaux
- En particulier, nos CPUs Cascade Lake ont 6 canaux
  - …mais nous n'avions installé qu'une barette par socket
  - Donc seulement **1/6 de bande passante** accessible…

Upgrade de serveur-paon4

- **Passage de 2 x 32 Go à 2 x 6 x 8 Go** pour utiliser les 6 canaux
- **Comparaison avec pc-bao2** (~ancienne config*) via STREAM
  - Bench d'opérations memory bound (ex : x[i] = a * y[i] + z[i])
  - Résultats en bande passante utile (ex : 3x taille pour Triad)
    - B.p. matérielle supérieure de 3/2x ou 4/3x (write-allocate)
  - Non optimisé NUMA → « pinning » sur socket avec taskset
  - Testé effets hyperthreading, AVX 256/512, nb coeurs…

* Pas tout à fait le même CPU (8 → 10 coeurs, 2.1 → 2,2 GHz), mais je ne m'attends pas à ce que ça ait une influence énorme sur les perf mémoire étudiées ici.

Résultats obtenus

- pc-bao2 (« avant ») : STREAM Triad à 13,4 Go/s par socket
  - Optimum pour 4 coeurs, sans hyperthreading, AVX 256-bit
  - Correspond à 17,8 Go/s matériel (**93 % b.p. théorique***)
- serveur-paon4 (« après ») : Triad à 51,0 Go/s (**3,8x mieux**)
  - Optimum pour 7 coeurs, sans hyperthreading, AVX 256-bit
  - Correspond à 68 Go/s matériel (**60 % b.p. théorique**)
  - Contacté Dell pour savoir si un tel écart est attendu

* Interface DDR4-2400 à 64 bits par transfert → 8 x 2,4 = 19,2 Go/s par canal attendu

Au programme

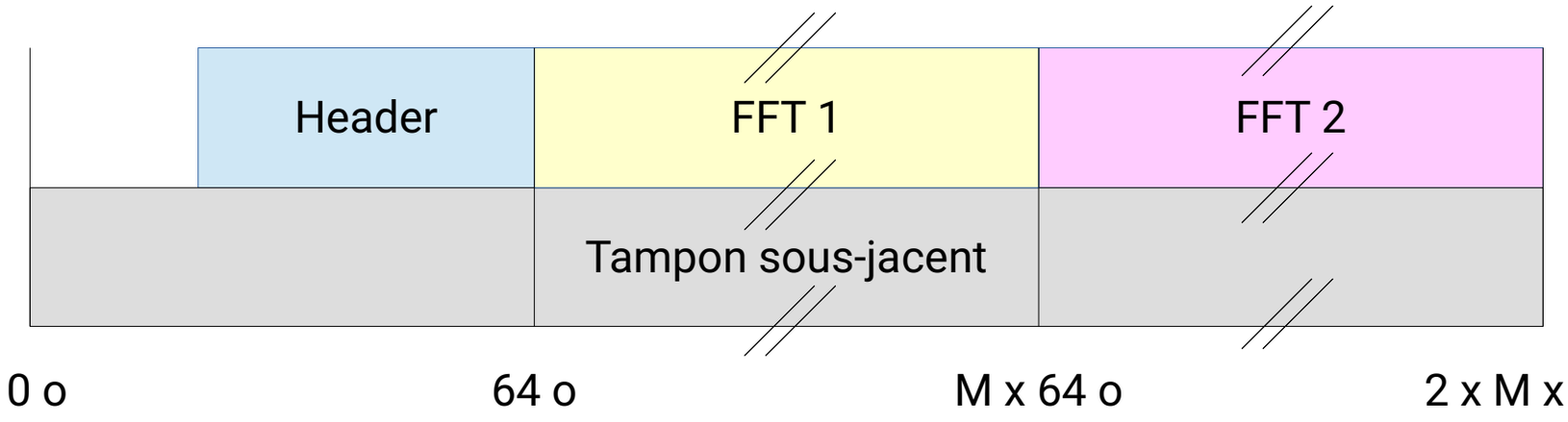
- Upgrade RAM serveur-paon4
- **Organisation des paquets de données**
- Intégration du nouveau corrélateur à TAcq
- Méthodes de travail git

Rappel de perf CPU

- Les CPUs sont optimisés pour des **accès mémoire linéaires**
  - Données rangées dans l'ordre où elles sont écrites/lues
- Ce rangement idéal n'est pas envisagé pour PAON-4
  - Il faudrait trier par flux → paquet → fréquence + du blocking
  - Pas favorable de réordonner si relu une seule fois derrière !
- Mais on peut **limiter la casse** des perfs à ~0,5x en adaptant un peu l'organisation fréquence → paquet → flux actuelle…

Alignement des données

- D’abord, aligner les données selon la **taille de vecteur SIMD**
  - 256-bit/32o pour AVX classique, 512-bit/64o pour AVX-512
  - Taille de FFT = 2^N grand donc 1ère FFT OK => toutes OK
  - Nécessite d’adapter les allocations de RacqMemZoneMgr
  - Pas *obligatoire* de changer la taille du header BRPaquet :



Parlons cache L1

- Les specs de cache L1d varient rarement. Pour Cascade Lake :
  - Capacité de 32 Kio
  - Décomposé en lignes de 64 octets
  - Ensembles avec une associativité de 8 (cf après)
  - On a donc 32 x 1024 / 64 / 8 = 64 ensembles
- CPU Intel actuel : capacité 48 Kio, associativité 12, reste idem
  - Pas un changement majeur pour ce qui suit

Problème d'associativité

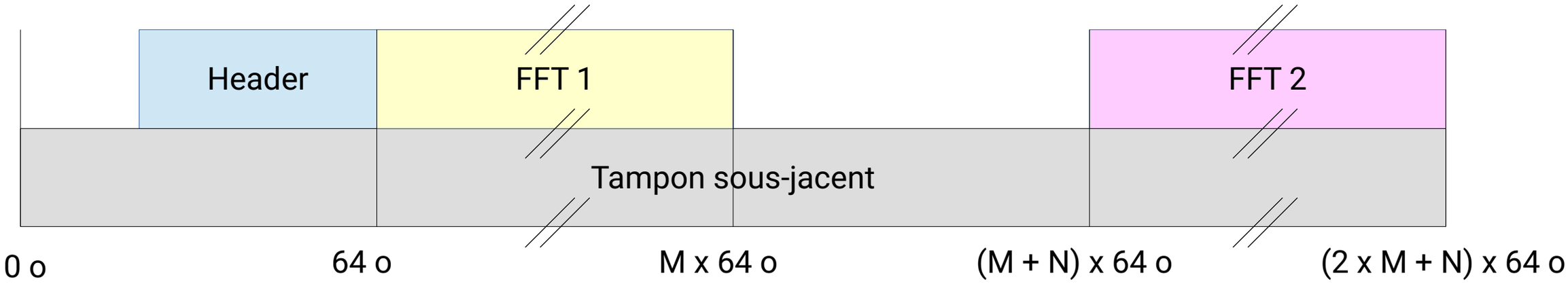
- Une adresse est associée à un **ensemble de lignes de cache**

Tag dans l'ensemble (associativité)	Choix ensemble	Intérieur de ligne
Bits 63 à 12	Bits 11 à 6	Bits 5 à 0

- Si les bits d'ensemble restent les mêmes, alors…
  - On utilise toujours le même ensemble (sur les 64 dispo)
  - Donc on a que 8 lignes de cache → **Capacité 1/64 = 512 o !**
  - Réduction du parallélisme matériel → **Moins de débit !**
- Se produit pour des adresses espacées de 2^N avec N grand
  - **Différentes FFTs à même fréquence** chez nous !

Solution : un peu de padding

- Mettre les FFTs successives (= paquets*) sur ≠ ensembles ?
  - Il suffit d'**insérer N lignes de cache vides** entre elles !



- On veut N ≥ 1 en AVX classique (256-bit) et N ≥ 2 en AVX-512
  - FFT entrelacées ⟶ On lit 2 vecteurs SIMD consécutifs

* Il y a un soucis similaire entre flux, mais on peut jouer sur le nombre de paquets à ce niveau

Au programme

- Upgrade RAM serveur-paon4
- Organisation des paquets de données
- **Intégration du nouveau corrélateur à TAcq**
- Méthodes de travail git

Etat actuel

- Préparé le terrain avec migration C++17 de Tacq
- Prochaines étapes : https://gitlab.in2p3.fr/baoradio/tacq/-/merge_requests/9
  - ~~Briques élémentaires de support SIMD~~ **=> Fait**
  - Alignement des allocations mémoire **=> En cours**
  - Padding entre FFTs dans les BRPaquet (comment ?)
  - Ajout du nouvel algo de corrélation & tests
- Freiné par un mois de novembre pénible, en train de reprendre

Au programme

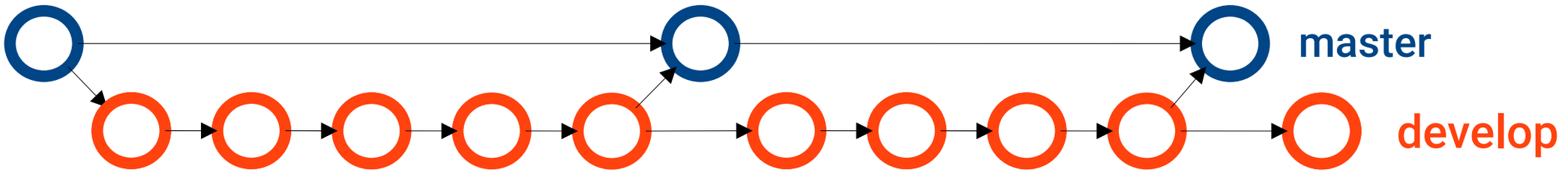
- Upgrade RAM serveur-paon4
- Organisation des paquets de données
- Intégration du nouveau corrélateur à TAcq
- **Méthodes de travail git**

Bénéfices et limites de CI

- Nous avons de l'intégration continue (CI) depuis cet été
  - Permet de détecter un certain nombre de problèmes
  - N'empêche pas de pousser ces problèmes sur master
  - Fonctionnera mal tant qu'ils ne sont pas résolus
- Pour cette raison et d'autres, on « protège » souvent master
  - Développement essentiellement fait sur d'autres branches
  - Processus d'intégration des changements contrôlé

Proposition : séparation master/develop

- Après discussion avec Réza…
  - Défavorable à une organisation 1 fonctionnalité = 1 branche
  - Ok pour définir une branche de développement distincte
  - Intégration occasionnelle à master, en « fast-forward » :



Implications

- Fusion vers master seulement via l'interface Gitlab
  - Accès en écriture depuis le client git local interdit
  - A la place, « merge request » develop → master régulière
  - Donne la possibilité de vérifier l'état CI avant de fusionner
  - Plus généralement, occasion de faire le point, màj la doc…
- Si OK, poussez vos développements locaux maintenant
  - Evitera de devoir les migrer sur la branche develop

Merci de votre attention !