Hétérogénéité pour atteindre une consommation énergétique proportionnelle dans les clouds

GreenDays@Rennes

Mardi 1er Juillet 2014

Violaine Villebonnet

Laurent Lefèvre – LIP Inria/ENS Lyon

Jean-Marc Pierson, Georges Da Costa, Patricia Stolf – IRIT Toulouse

Hétérogénéité pour atteindre une consommation énergétique proportionnelle dans les clouds

- Pourquoi la proportionnalité énergétique ?
- Approches actuelles homogènes
- Processeur multi-cœurs hétérogènes
- Notre approche : hétérogénéité au sens large
- Challenges techniques :
  - Migrations entre architectures hétérogènes
- Premiers résultats
- Futures directions

Pourquoi la proportionnalité ?

Grande définition du concept en 2007:

*Luiz André Barroso et Urs Hölzle, « **The case for Energy-Proportional Computing** », IEEE Computer*

<u>Au niveau serveur :</u>

Importante consommation des serveurs « en veille »

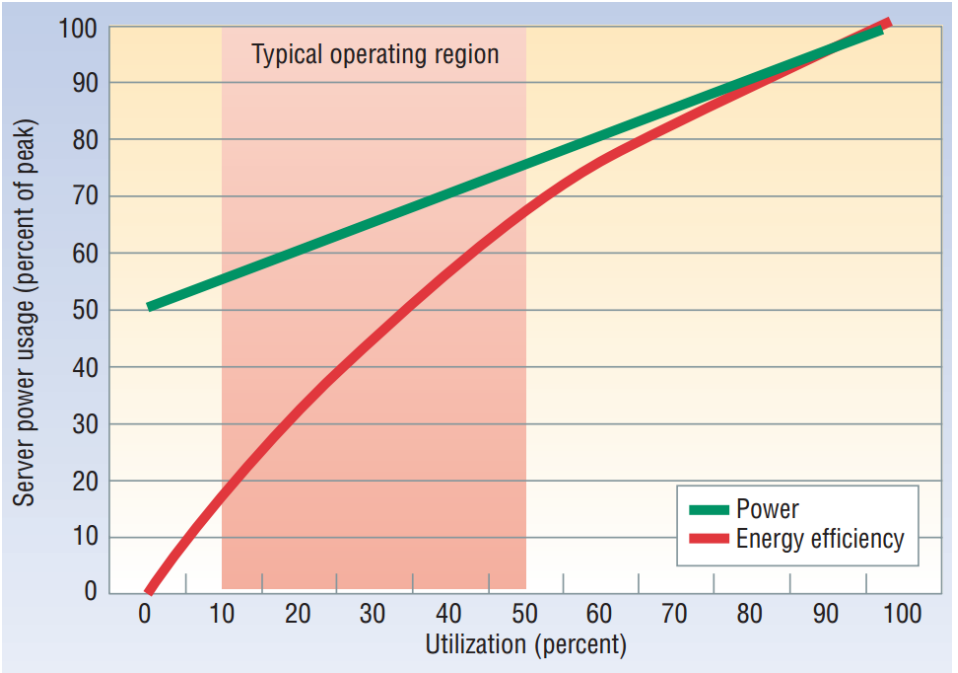
Charge moyenne des serveurs d'un datacenter

entre 10 et 50 % → très inefficace

<u>Au niveau réseau :</u>

Encore moins de proportionnalité

Consommation des switchs quasi constante



Consommation et efficacité énergétique d'un serveur en fonction de son utilisation

Approches actuelles : architectures homogènes

- Au niveau architecture :
  - DVFS : Dynamic Voltage and Frequency Scaling

  Adapte la fréquence du processeur à la puissance de calcul nécessaire

  - RAPL : Running Average Power Limit

  Introduit par Intel dans ses processeurs Sandy Bridge
  Permet de limiter la consommation moyenne du processeur et autre composants de la carte mère sur une certaine durée

- Au niveau datacenter :
  - Dynamic right-sizing

  Seuls les serveurs utilisés sont allumés, les autres sont éteints, ou en mode basse consommation
  On démarre, ou réveille, les serveurs selon la demande

Processeur multi-cœurs hétérogènes

ARM big.LITTLE

2 processeurs (4 cœurs chacun) :

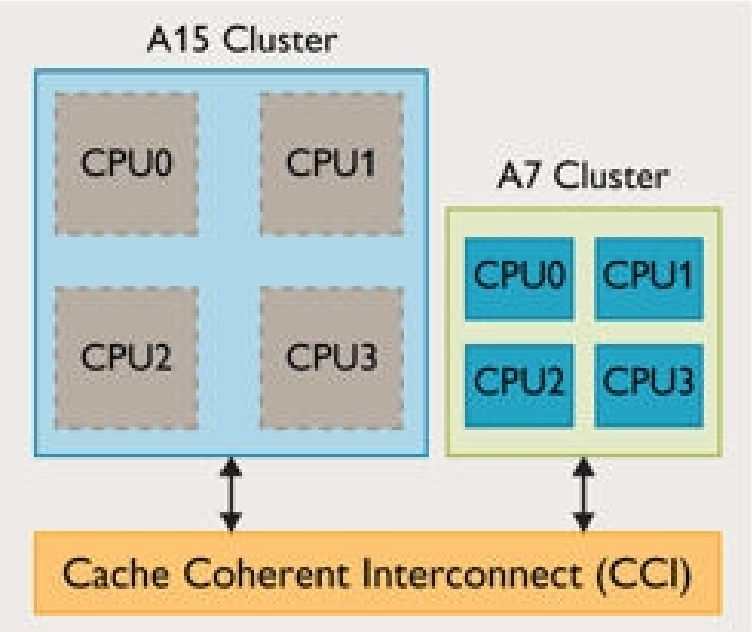
- LITTLE (Cortex A7)
- big (Cortex A15)

Interconnectés par un système de cache cohérence

BUT→ Prolonger la longévité de la batterie des appareils mobiles qui sont la majorité du temps en veille

3 modes d'utilisation :

- Cluster migration (4 / 4)
- In-kernel switcher (4 * (1 / 1))
- Global Task Scheduling (8)



big.LITTLE « Cluster migration »

Notre approche : Hétérogénéité plus large

A l'échelle d'un datacenter → ARM ne suffit pas !
On a besoin de réelles performances pour répondre aux pics de charge.

Notre approche :

Processeurs basse-consommation pour les moments de calme

et

Serveurs classiques quand on a besoin de performance

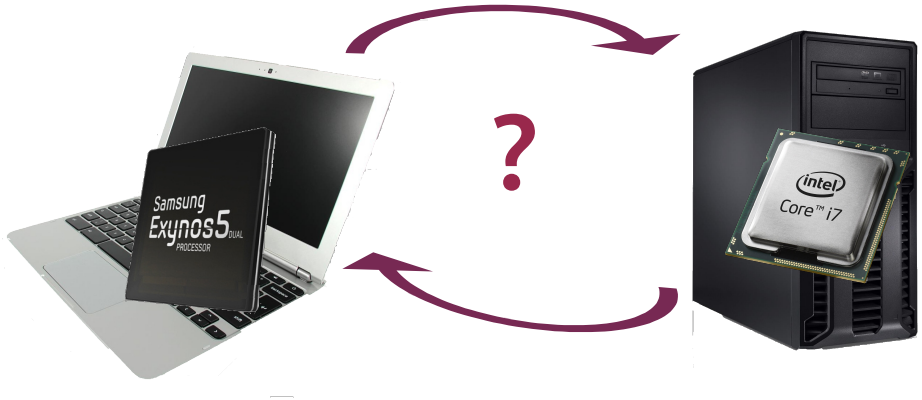
→ réduire les coûts statiques

→ utiliser les serveurs classiques seulement à leur niveau d'utilisation le plus efficace énergétiquement

+ autres techniques : DVFS, switch off/on,… pour améliorer la linéarité de la consommation

Challenges techniques

- Différentes architectures : ARM et x86
- Différentes machines et non pas processeurs :
  Pas de système de cache cohérence



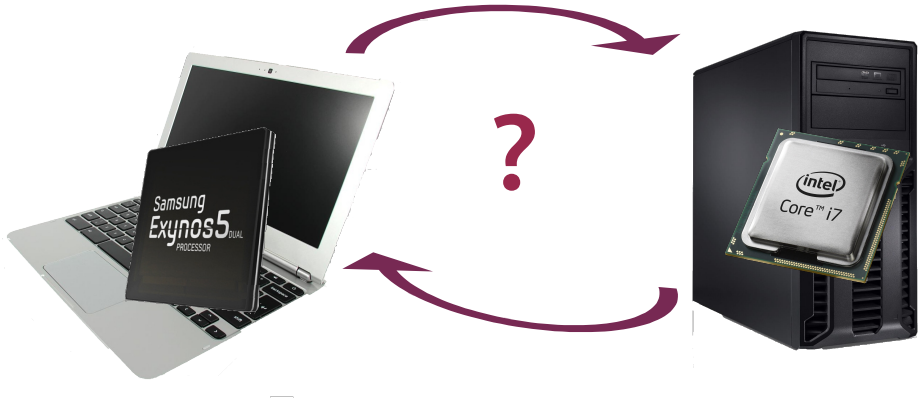
Comment les rendre « compatibles » et pouvoir passer facilement d'une architecture à l'autre ?

- live migration sans conséquence lourde pour l'application en cours d'exécution

- migration la plus rapide possible

Challenges techniques

- Différentes architectures : ARM et x86
- Différentes machines et non pas processeurs :
  Pas de système de cache cohérence



Comment les rendre « compatibles » et pouvoir passer facilement d'une architecture à l'autre ?

- live migration sans conséquence lourde pour l'application en cours d'exécution

- migration la plus rapide possible

→ Première idée Approche cloud classique : Machine virtuelle

2 architectures physiques → 2 choix d'architecture pour la machine virtuelle

Quand la VM n'est pas sur la bonne architecture physique, on utilise l'émulation avec QEMU

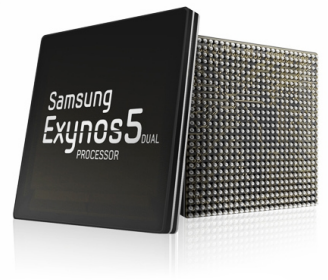
→ Quel est le surcoût de l'émulation ?

→ Quelle architecture choisir pour la VM ?

Architectures ARM et virtualisation

Certains processeurs ARM récents disposent d'extensions de virtualisation

Seulement 2 : Cortex-A7 et Cortex-A15



- Nvidia Tegra 4 (4 cœurs Cortex-A15)

Tablette Microsoft Surface 2, Tegra Note 7

- Samsung Exynos 5250 (2 cœurs Cortex-A15)

Tablette Nexus 10, Samsung Chromebook 303

Installation d'un système Linux : Debian, Ubuntu,..

KVM/ARM introduit dans noyau Linux 3.9

(aujourd'hui version 3.15)

Coûts d'émulation – Premiers résultats

- Comparaison du surcoût d'émulation sur 2 architectures :
  - ARM : Samsung Chromebook (2 processeurs ARM Cortex-A15)
  - X86 : Dell PowerEdge R720 (2 processeurs Intel Xeon 6 cœurs)

- Qemu-user : exécution de binaires par traduction d'instructions
- Benchmark nbench : calcul entier/flottant

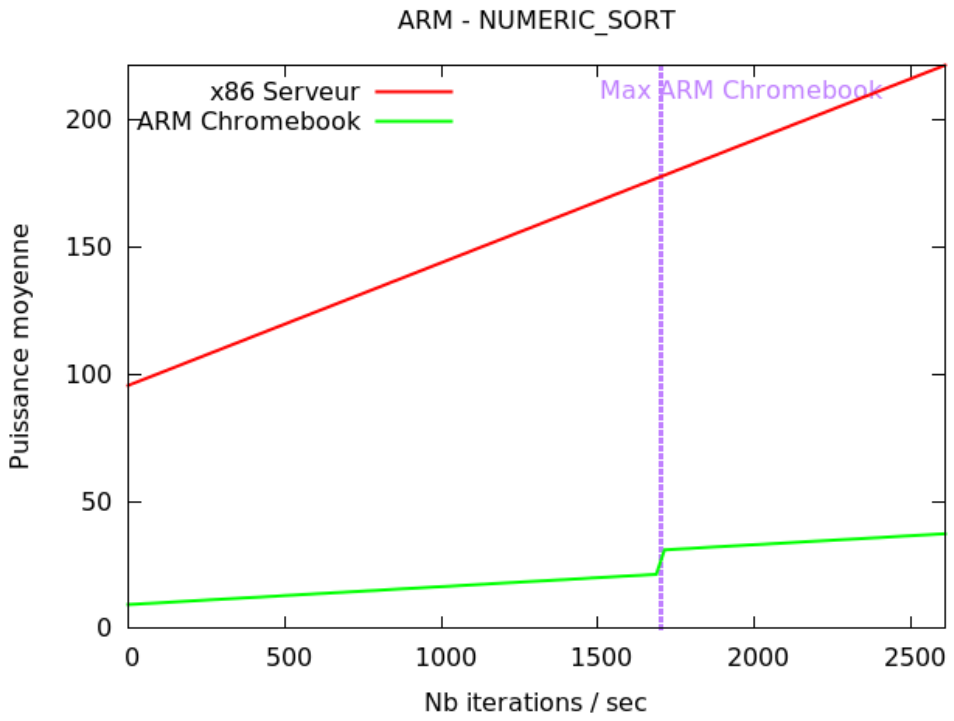
→ Même surcoût d'émulation :

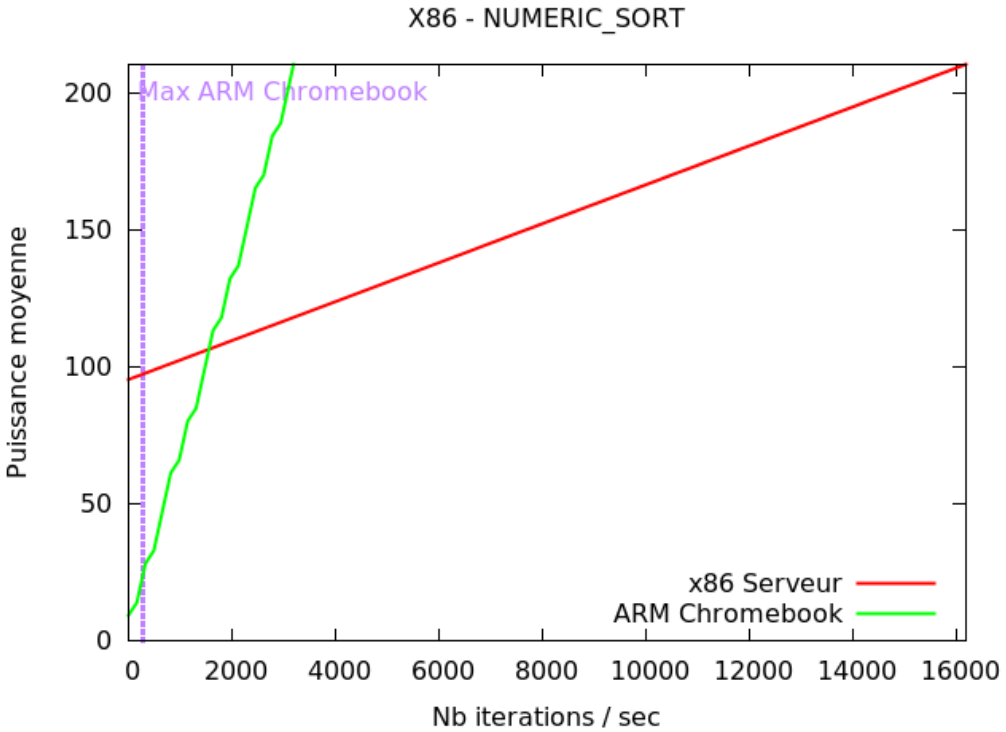
  - Calcul entier : 6,5 fois plus lent
  - Calcul flottant : 30 à 40 fois plus lent

Comparaison d'architecture VM – Premiers résultats

- VM ARM :
  Natif sur le processeur ARM
  Émulation sur le processeur x86

- VM x86 :
  Natif sur le processeur x86
  Émulation sur le processeur ARM



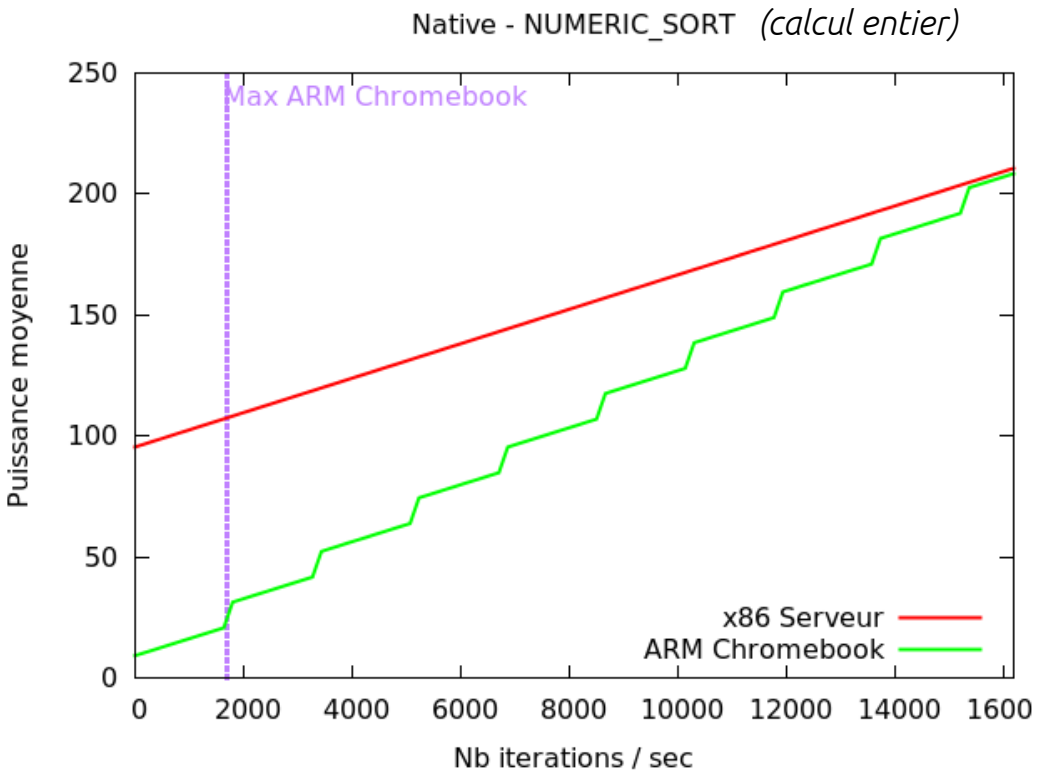


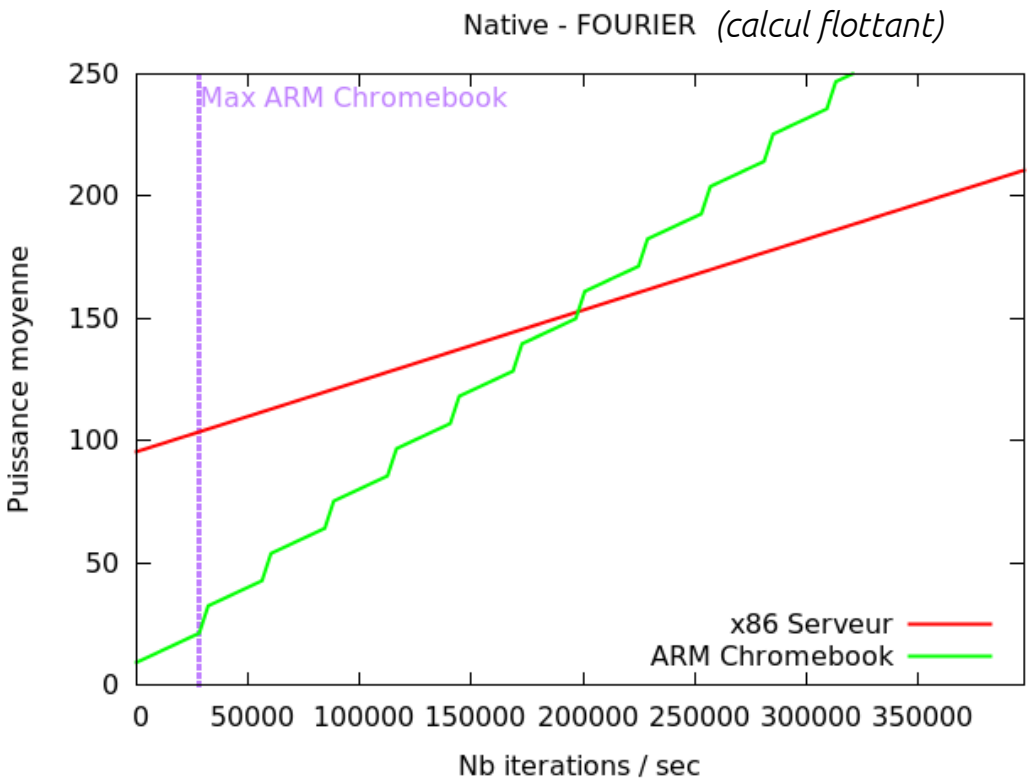
Itérations/sec maximum : ARM → 2600 x86 → 16170

→ VM ARM limite énormément la performance globale
→ VM x86 apporte le meilleur compromis

Comparaison performances natives – Premiers résultats

- En profitant des performances natives de chaque architecture, on se rapproche de la proportionnalité
- Mais pour pouvoir en profiter, tout dépend du type d'applications



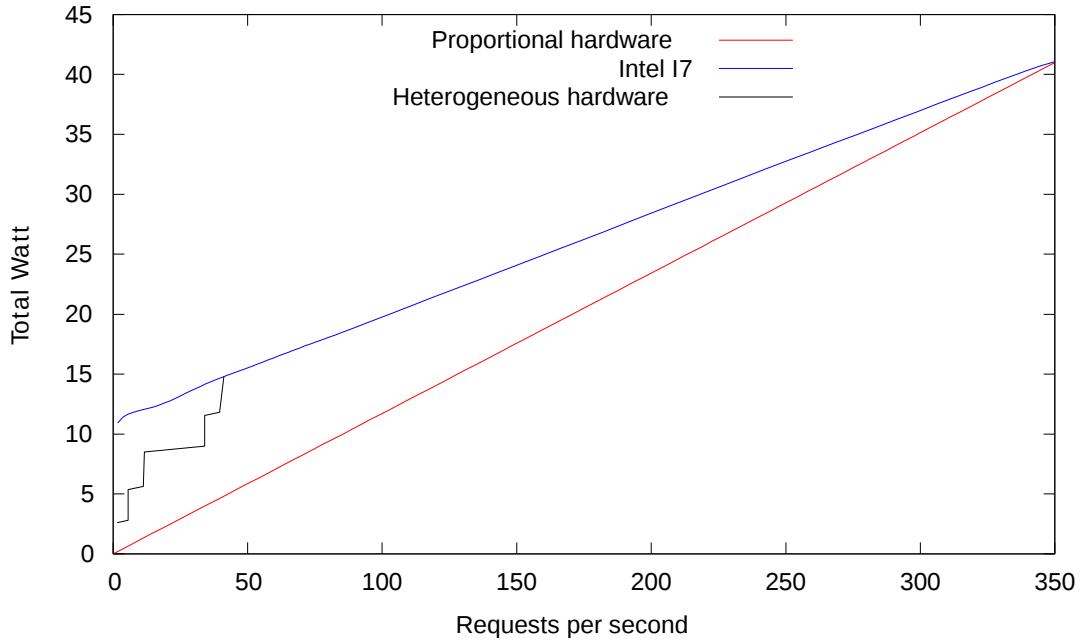


Performance native Serveur / Chromebook	~9,5	14
Nombre de Chromebook pour atteindre la performance d'un serveur x86	10	7

Exemple des applications sans état

Cas d'un serveur web sans état

Facilement l'éteindre et le rallumer sur une autre architecture en fonction du nombre de requêtes



Georges Da Costa,
*« **Heterogeneity: the key to achieve Power-Proportional Computing** », CCGrid 2013*

→ Mais pour les autres types d'applications ?

Dépend de leur propriété de parallélisation, la taille des données à transférer en cas de migrations...

Futures directions

- Mieux cibler les différents types d'applications et les actions à entreprendre

- Pour profiter des performances natives :
  - Checkpoint/Restart d'applications
  - Système de cache cohérence via mémoire virtuelle partagée entre archis hétérogènes

- Au delà du challenge technique, il faut ensuite savoir comment utiliser ce levier de migrations entre architectures !

Comment prédire l'évolution des besoins en ressources d'une application, si ce changement est durable et va mériter une migration, etc....

- Autre leviers ?

Futures directions

- Mieux cibler les différents types d'applications et les actions à entreprendre

- Pour profiter des performances natives :
  - Checkpoint/Restart d'applications
  - Système de cache cohérence via mémoire virtuelle partagée entre archis hétérogènes

- Au delà du challenge technique, il faut ensuite savoir comment utiliser ce levier de migrations entre architectures !

Comment prédire l'évolution des besoins en ressources d'une application, si ce changement est durable et va mériter une migration, etc....

- Autre leviers ?

Des questions, des remarques :

violaine.villebonnet@irit.fr