RETOUR SUR LE CLOUD CHALLENGE FRANCE GRILLES

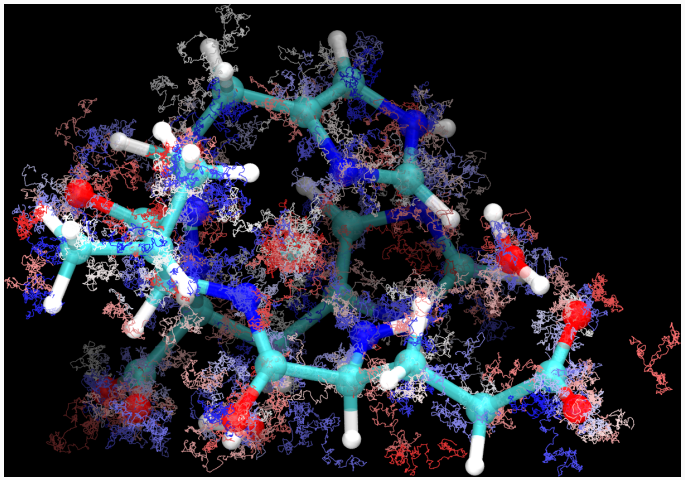
Utilisation HPC en Chimie Quantique

Anthony Scemama[1], Thomas Applencourt[1], Michel Caffarel[1], Georges Da Costa[2]

4/11/2015

[1]Lab. Chimie et Physique Quantiques, IRSAMC, UPS/CNRS, Toulouse
[2]IRIT, Toulouse

Description quantitative de systèmes chimiques complexes (différences d'énergies)



Applications scientifiques et technologiques:

- Industrie pharmaceutique : Drug design
- Électronique : Nano- et micro-electronique
- Matériaux : Nanotubes de carbone, graphène, *etc*
- Catalyse : Réactions enzymatiques, industrie du pétrole

Résolution de l'équation de Schrödinger:

$$\mathcal{H}\Psi(\mathbf{R}) = -\frac{1}{2}\nabla^2\Psi(\mathbf{R}) + V(\mathbf{R}) = E_0\Psi(\mathbf{R})$$

Ψ : Fonction d'onde électronique de l'état fondamental du système

$\mathbf{R}$: Vecteur de $\mathbb{R}^{3N}$ contenant les positions des électrons

E_0 : Énergie correspondante

C'est une EDP dans un espace à $3N$ dimensions !

Approches usuelles

- $\Psi(\mathbf{R})$ est exprimée sur base *finie* de fonctions
- On résout le problème *approché* avec des méthodes de Krylov $\Rightarrow$ Algèbre linéaire avec de grandes matrices ($\sim 10^8 \times 10^8$)

Approches usuelles

- $\Psi(\mathbf{R})$ est exprimée sur base *finie* de fonctions
- On résout le problème *approché* avec des méthodes de Krylov $\Rightarrow$ Algèbre linéaire avec de grandes matrices ($\sim 10^8 \times 10^8$)

Notre approche

1. CIPSI : Résoudre le problème approché dans un espace plus petit ($\sim 10^4 \times 10^4$)
2. QMC : Utiliser des méthodes stochastiques (**Monte Carlo Quantique**) pour résoudre le problème *exact* dans tout le reste de l'espace

Pour quantifier la qualité de notre approche : Benchmark

Énergies d'atomisation[1] de 55 molécules

Be, CH_3Cl, F, HCO, Na, P, $SiH_2(^3B_1)$, BeH, CH_4, F_2, HF, Na_2, P_2, SiH_3, C, Cl, H_2CO, HOCl, NaCl, PH_2, SiH_4, C_2H_2, Cl_2, H_2O, Li, NH, PH_3, SiO, C_2H_4, ClF, H_2O_2, Li_2, NH_2, S, SO, C_2H_6, ClO, H_2S, LiF, NH_3, S_2, SO_2, CH, CN, H_3COH, LiH, NO, Si, $CH_2(^1A_1)$, CO, H_3CSH, N, O, Si_2, $CH_2(^3B_1)$, CO_2, HCl, N_2, O_2, Si_2H_6, CH_3, CS, HCN, N_2H_4, OH, $SiH_2(^1A_1)$

[1] $\mathrm{AE}(\mathrm{H_2O}) = E(\mathrm{H_2O}) - [2 \times E(\mathrm{H}) + E(\mathrm{O})]$

1. Un grand nombre de calculs indépendants (55 molécules)
   - Runs CIPSI
   - 16 cœurs à mémoire partagée (500 – 5 000 heures CPU/job)
2. De grands calculs distribués
   - Runs QMC
   - Chaque job utilise des dizaines de milliers de cœurs (5 000 – 200 00 heures CPU/job)

Machines sur lesquelles tournent nos codes

- **Curie** (TGCC/CEA/Genci) : 5000 2x8-core nodes (Sandy bridge)
- **Occigen** (Cines) : 2100 2x12-core nodes (Haswell)
- **Eos** (Calmip) : 600 2x10-core nodes (Ivy-bridge)
- **Cluster** du LCPQ : 35 2x16 core nodes (Sandy/Ivy/Haswell/Nehalem/Atom/AMD)

MOTIVATIONS POUR LE CLOUD

- Dans un futur proche : supercalculateurs avec des millions de cœurs
- La résilisence doit être prise en compte
- Les implémentations fortement couplées (MPI) vont avoir du mal à passer à l'échelle
- Les algorithmes à faible couplage sont de bons candidats

MOTIVATIONS POUR LE CLOUD

- Dans un futur proche : supercalculateurs avec des millions de cœurs
- La résilisence doit être prise en compte
- Les implémentations fortement couplées (MPI) vont avoir du mal à passer à l'échelle
- Les algorithmes à faible couplage sont de bons candidats

1. Ces motivations sont les mêmes que celles qui motivent le calcul sur grille
2. La production n'est pas constante toute l'année : plus de flexibilité pour les utilisateurs
3. Peut-être moins consommateur d'énergie pour nos codes

BENCHMARKS CIPSI

Code CIPSI (Quantum Package)

- Pour le moment, OpenMP
- Memory-bound : efficace sur CPUs à faible consommation
- Point chaud : instruction `popcnt` (hardware depuis SSE4.2)

Code CIPSI (Quantum Package)

ID	CPU	GHz	Cœurs	Cache	Lieu
CALMIP	Intel Xeon E5-2680 v2	2.8	2x10	25 MiB	CALMIP
DESK	Intel Xeon E3-1271 v3	3.6	1x4	8 MiB	Desktop
MOON	Intel Atom C2730	2.4	1x8	1 MiB	Moonshot
IPHC-NHM	Westmere (E/L/X)56xx (E/L/X)56xx (Nehalem-C)	2.5	2x8	4 MiB	IPHC Cloud
IPHC-SNB	Intel Xeon E312xx Sandy Bridge	2.6	2x8	4 MiB	IPHC Cloud
LAL	QEMU Virtual CPU (cpu64-rhel6)	2.7	1x8	4 MiB	LAL Cloud

- CALMIP : Meso-centre Midi-Pyrénées
- IPHC : Institut Pluridisciplinaire Hubert Curien, Strasbourg (Openstack)
- LAL : Laboratoire de l'Accélérateur Linéaire, Orsay (Stratuslab)

Exécution mono-thread

Haswell CPU @ 3.6GHz	
SSE2 gfortran	1040.1 s
SSE2 ifort	687.0 s
>=SSE4.2 ifort	122.0 s

- Dépend du compilateur
- Les instructions >= SSE4.2 (`popcnt` hardware) sont essentielles

			@ 1 GHz
LAL	SSE2	1193.0 s	3180.5 s
MOON	SSE4.2	375.9 s	902.2 s
IPHC-NHM	SSE4.2	195.4 s	488.5 s
CALMIP	AVX	160.3 s	448.8 s
IPHC-SNB	AVX	157.9 s	(!) 410.5 s
DESK	AVX2	122.0 s	439.2 s

EXÉCUTION MULTI-THREAD

LAL	8	SSE2	153.7 s
MOON	8	SSE4.2	60.0 s
DESK	4	AVX2	31.1 s
DESK	8(HT)	AVX2	24.1 s
IPHC-SNB	8	AVX	21.2 s
IPHC-NHM	16	SSE4.2	19.0 s
CALMIP	20	AVX	9.7 s
CALMIP	40(HT)	AVX	9.3 s

- Le code multi-thread fonctionne bien sur les VMs : $\times$10.3/16 cores, $\times$7.4/8 cores
- L'IPHC propose des VMs avec des performances comparables à CALMIP

Conclusion des benchmarks

Les performances des VMs pour le calcul multi-threadé sont excellentes **à condition d'avoir un accès complet au CPU** (`host-passthrough`).

BENCHMARKS QMC

code QMC (QMC=CHEM)

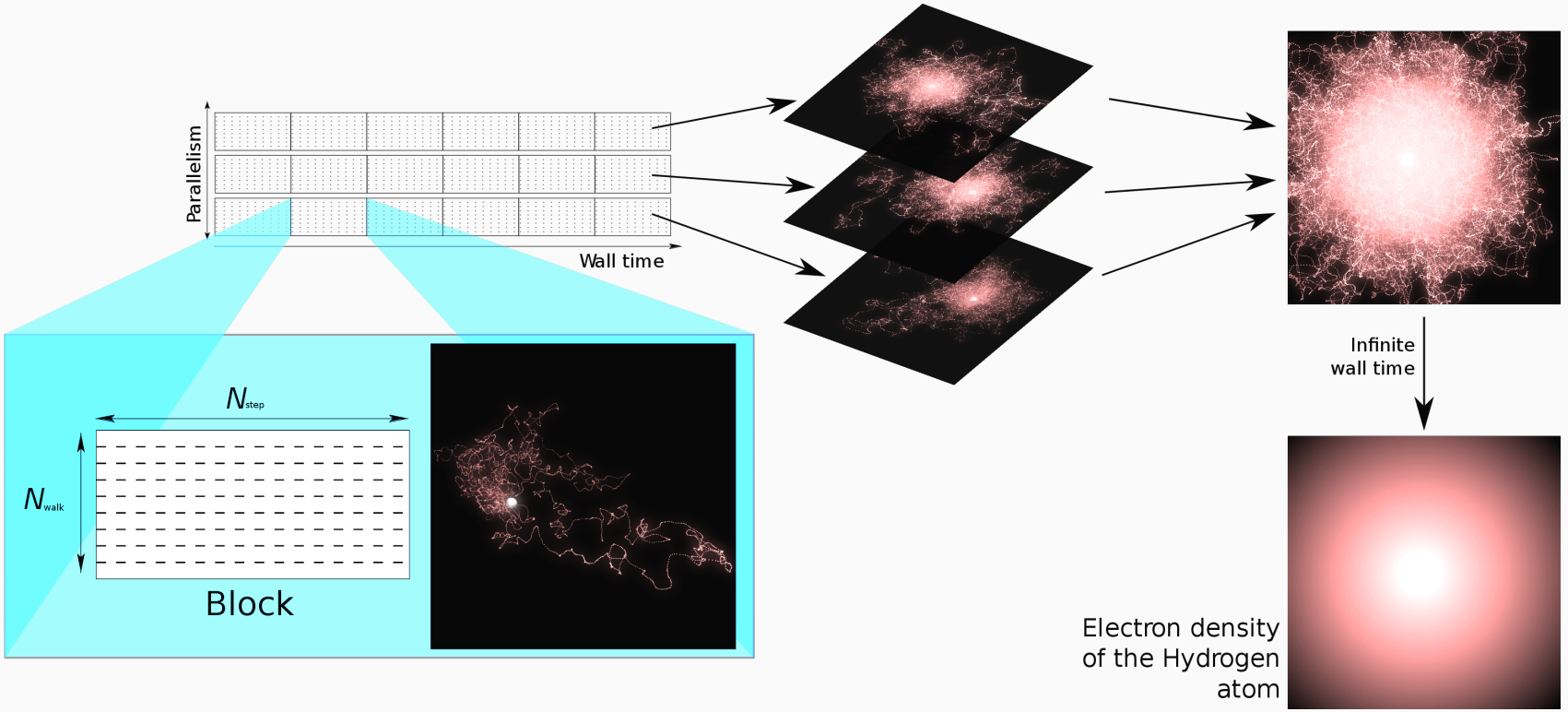
- Implémentation client/server : totalement asynchrone
- $<$100 MiB par coeur, 2.1 MiB pour le benchmarks
- CPU/cache-bound
- Peut utiliser autant de CPUs que possible : 76 800 cœurs sur Curie en 2011 (0.96 PFlops/s pendant 24h).

Code QMC (QMC=Chem)

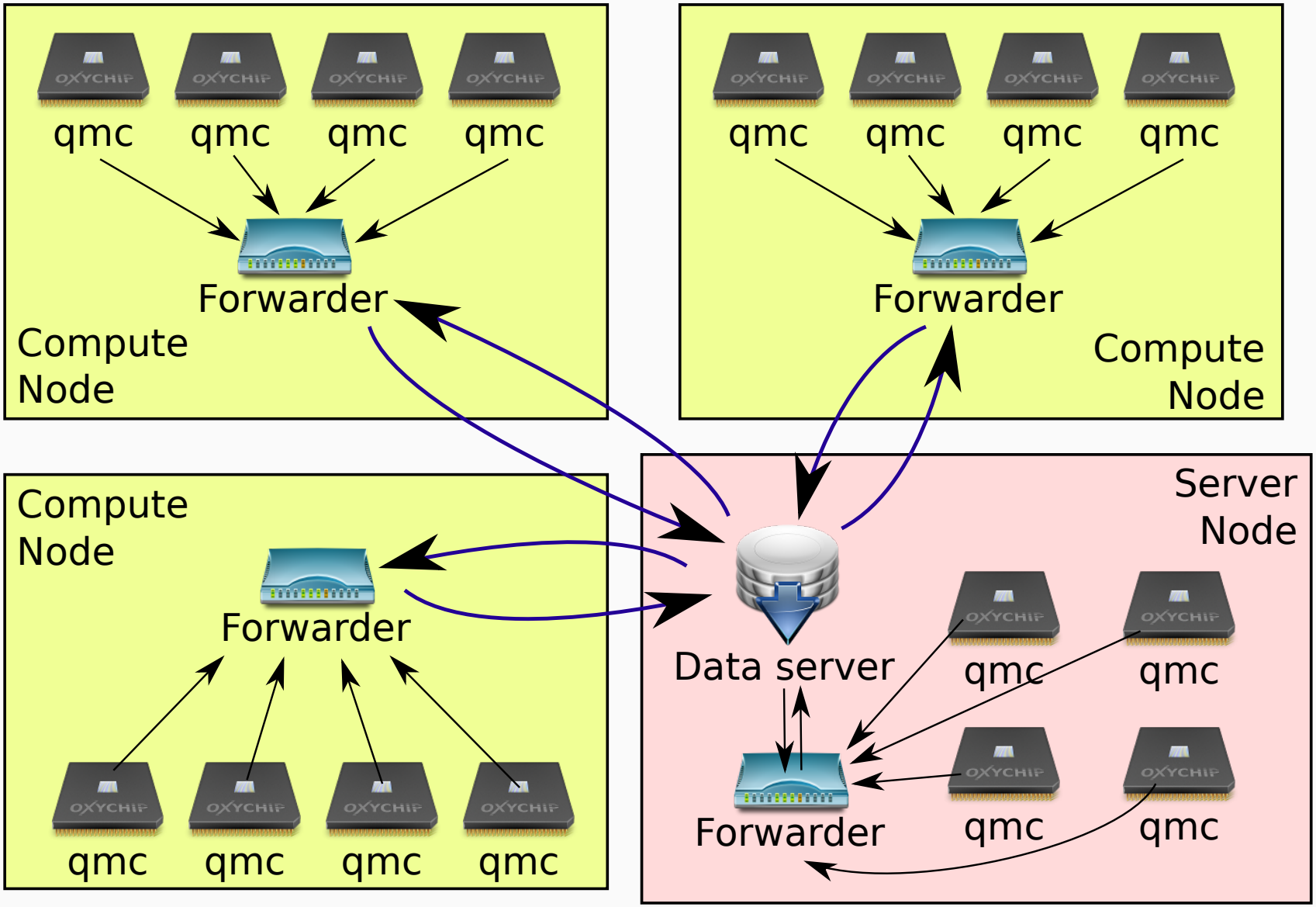
- Implémentation client/server : totalement asynchrone
- $<$100 MiB par coeur, 2.1 MiB pour le benchmarks
- CPU/cache-bound
- Peut utiliser autant de CPUs que possible : 76 800 cœurs sur Curie en 2011 (0.96 PFlops/s pendant 24h).

- Processus de calcul : Fortran mono-thread
- Forwarder/Data server : OCaml
- Communications : Bibliothèque ØMQ
- Résilience : tout processus peut être tué sans affecter le reste de la simulation.

CODE QMC (QMC=CHEM)



CODE QMC (QMC=CHEM)



Exécution mono-cœur (secondes)

CPU	GHz	SSE2	SSE4.2	AVX	AVX2
MOON	2.4	43.01	42.01		
IPHC-NHM	2.5	19.12	18.87		
LAL	2.7	17.80			
IPHC-SNB	2.8	15.56	14.82	14.00	
CALMIP	2.8	15.39	14.51	13.78	
DESK	3.6	10.30	9.52	8.55	8.21

- 2 MiB : sort du cache Atom
- Sensible à la fréquence
- Amélioration avec le jeu d'instructions
- LAL : bonnes performances malgré SSE2

Exécution mono-cœur (secondes)

Renormalisé @ 1GHz:

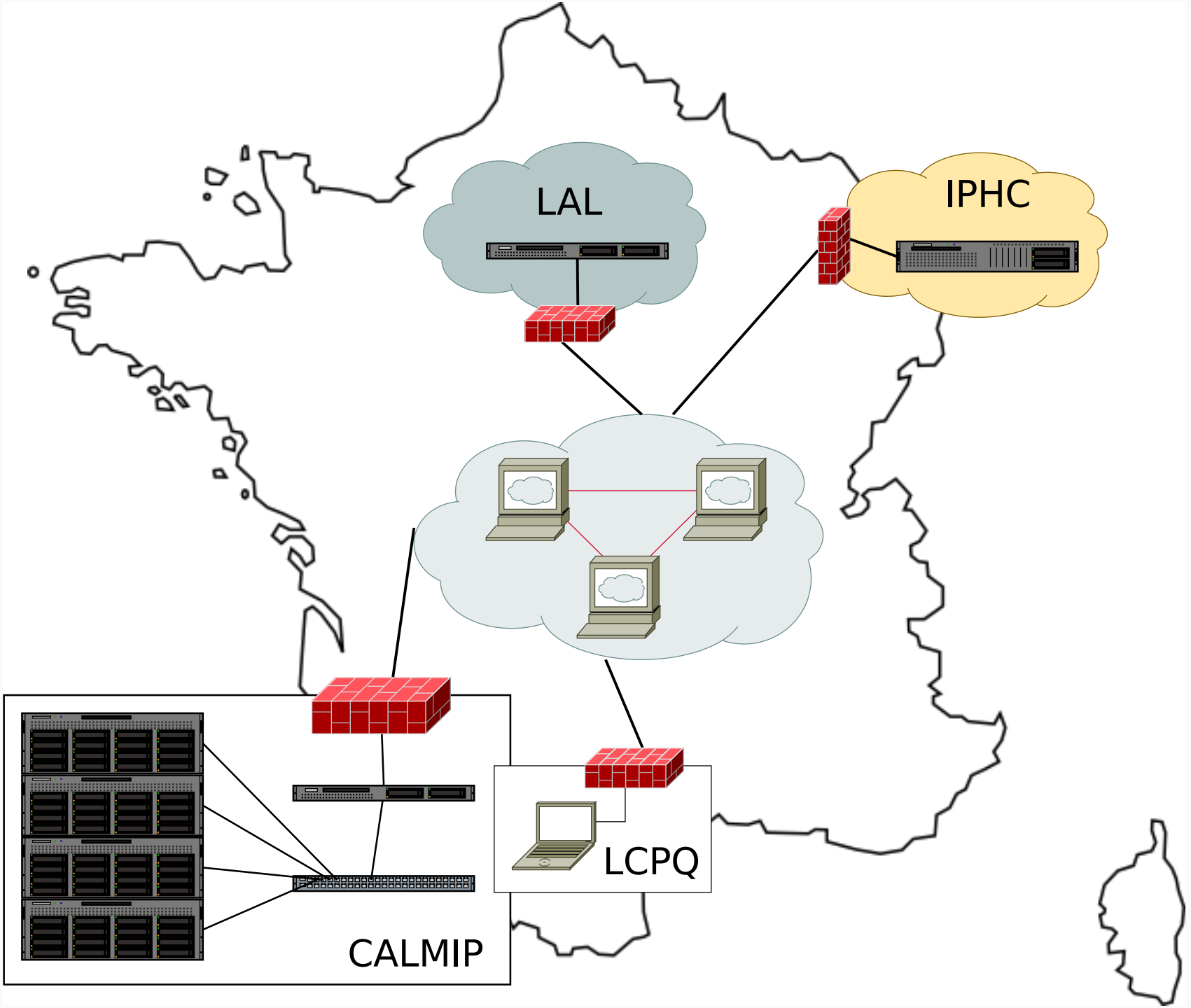
CPU	SSE2	SSE4.2	AVX	AVX2
MOON	103.22	100.82		
IPHC-NHM	47.80	47.16		
LAL	47.45			
IPHC-SNB	40.46	38.53	(!) 36.40	
CALMIP	43.09	40.63	38.54	
DESK	37.15	34.27	30.78	29.56

- SSE2 plus rapide sur CALMIP que IPHC-NHM : MKL >SSE2
- SSE2 plus rapide sur DESK que CALMIP : Bande passante Haswell
- IPHC-SNB plus rapide que CALMIP : turbo actif? Haswell déguisé?

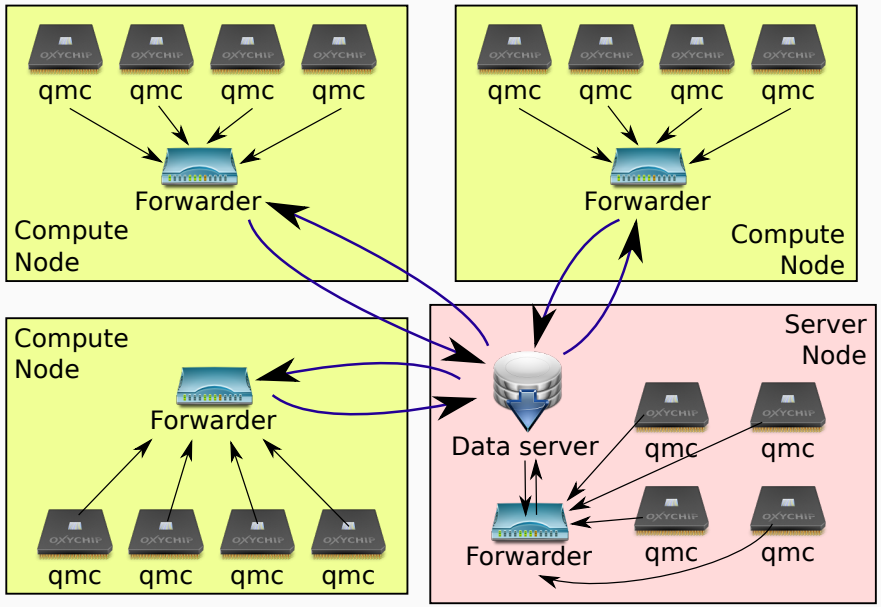
Conclusion des benchmarks

Les performances des VMs sont bonnes pour les VMs généralistes, et excellentes pour les VMs qui exposent le CPU.

Application : calcul distribué utilisant meso-centre et cloud


LAL
IPHC
LCPQ
CALMIP

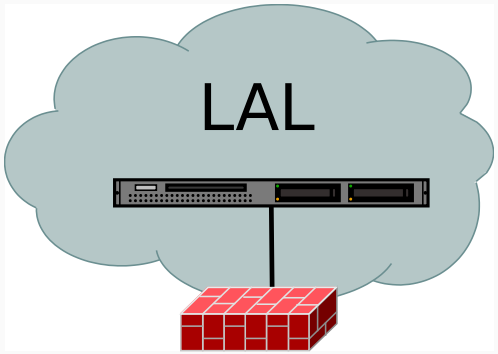
Déroulement d'un calcul



- `qmcchem -d` : Dataserver
- `qmcchem -q` : Forwarder / QMC

```
$ qmcchem run zn.ezfio
Scheduler : SLURM
Launcher  : srun
  25278 : qmcchem run -d zn.ezfio
Server address: tcp://130.120.229.139:41578
 25314 : srun qmcchem run -q tcp://130.120.229.139:41578 zn.ezfio
```

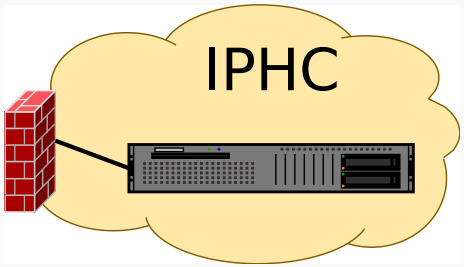
1. Démarrage d'un serveur au LAL



- Démarrer une VM à un cœur
  134.158.75.78
- Uploader QMC=Chem et l'input
- Démarrer un run à 1 cœur:
  Dataserver + Forwarder + QMC
- Récupérer le numéro de port
  34298

```
root@lal:~# qmcchem run zn.ezfio
Scheduler : Batch
Launcher  : env
  4172 : qmcchem run -d zn.ezfio
Server address: tcp://134.158.75.78:34298
 4193 : env qmcchem run -q tcp://134.158.75.78:34298 zn.ezfio
```

2. Démarrage d'un client à l'IPHC

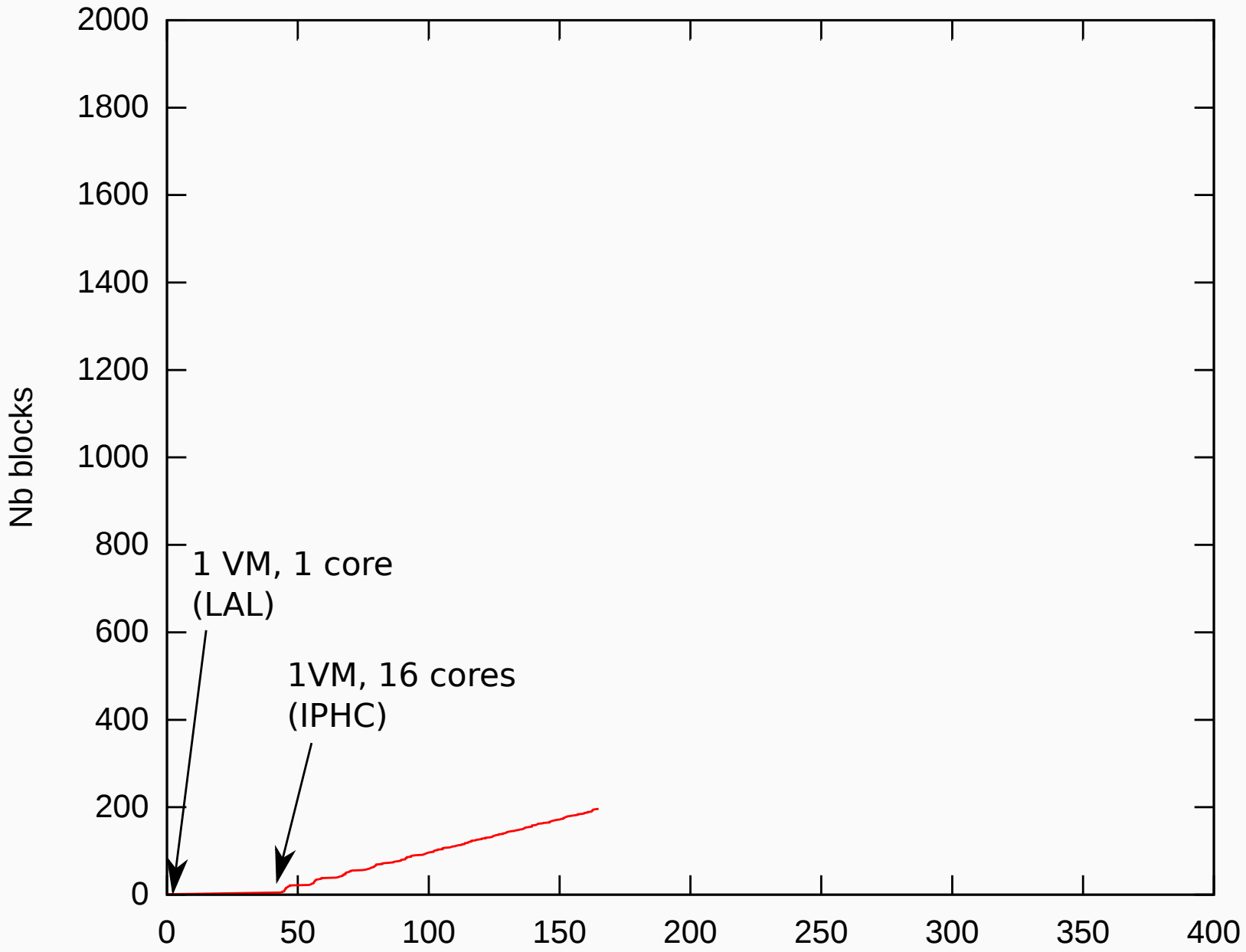


- Démarrer une VM à un 16 cœurs
- Uploader la clé privée SSH
- Uploader QMC=Chem et l'input
- Démarrer un proxy ØMQ ↔ SSH:
  localhost:34308 ↔ 134.158.75.78:34298
- Démarrer un client à 16 cœurs, connecté au proxy : 1 Forwarder + 16 QMC

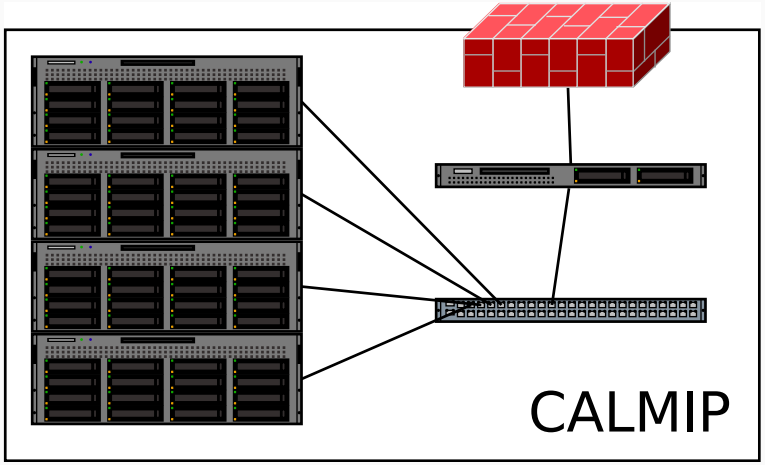
```
root@iphc:~# qmc_proxy.py tcp://134.158.75.78:34298 root@134.158.75.78
Proxy : tcp://iphc:34308

root@iphc:~# for i in {0..15}
> do
>  taskset -c $i qmcchem run -q tcp://localhost:34308 zn.ezfio &
> done ; wait
```

2. Démarrage d'un client à l'IPHC



3. Démarrage d'un job à CALMIP

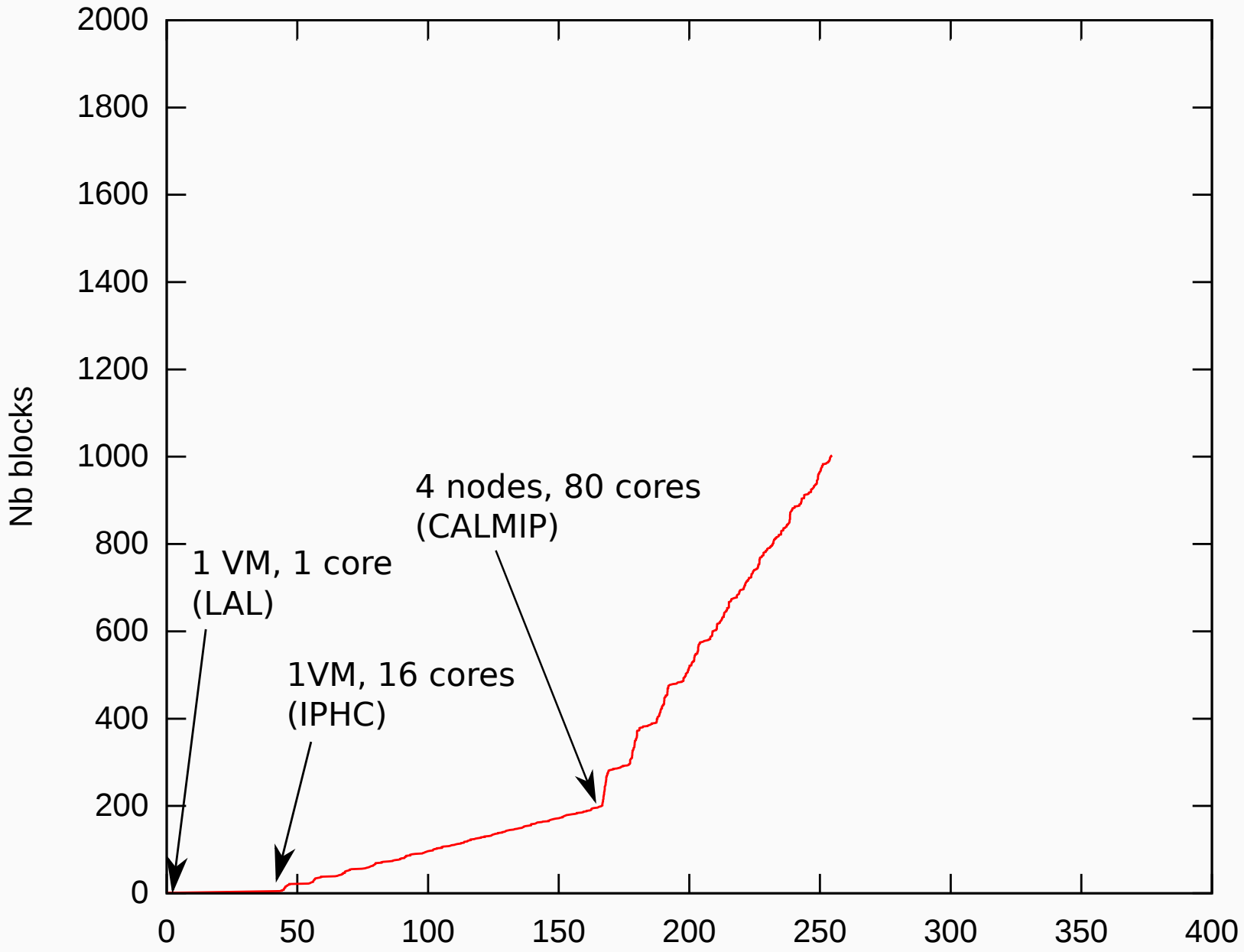


- Ma clé privée SSH est déjà là
- Démarrer un proxy ØMQ ↔ SSH sur le nœud de login:
  eoslogin1:34308 ↔ 134.158.75.78:34298
- soumettre un job à 80 cœurs, connecté au proxy :
  4 Forwarders + 80 QMC

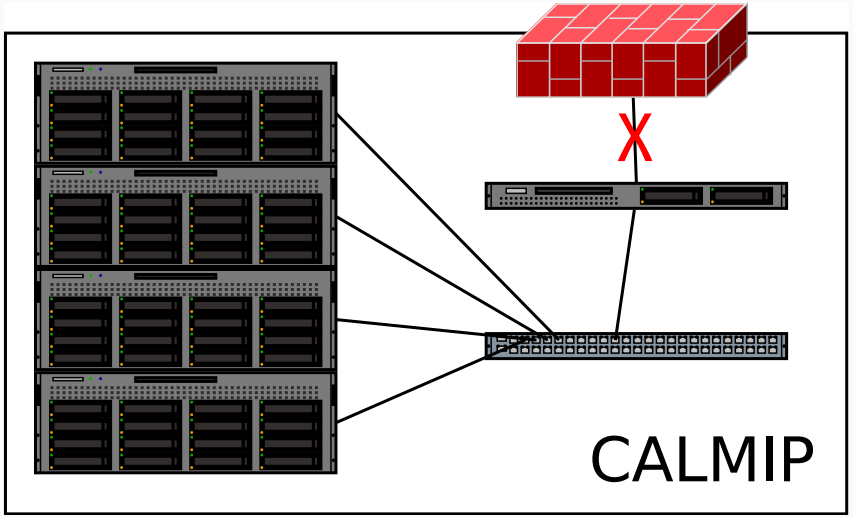
```
scemama@eoslogin1 $ qmc_proxy.py  tcp://134.158.75.78:34298 \
  root@134.158.75.78
Proxy : tcp://eoslogin1:34308

scemama@eoslogin1 $ cat << EOF | sbatch -N 4 -n 80
#!/bin/bash
srun qmcchem run -q tcp://eoslogin1:34308  zn.ezfio
EOF
```

3. DÉMARRAGE D'UN JOB À CALMIP

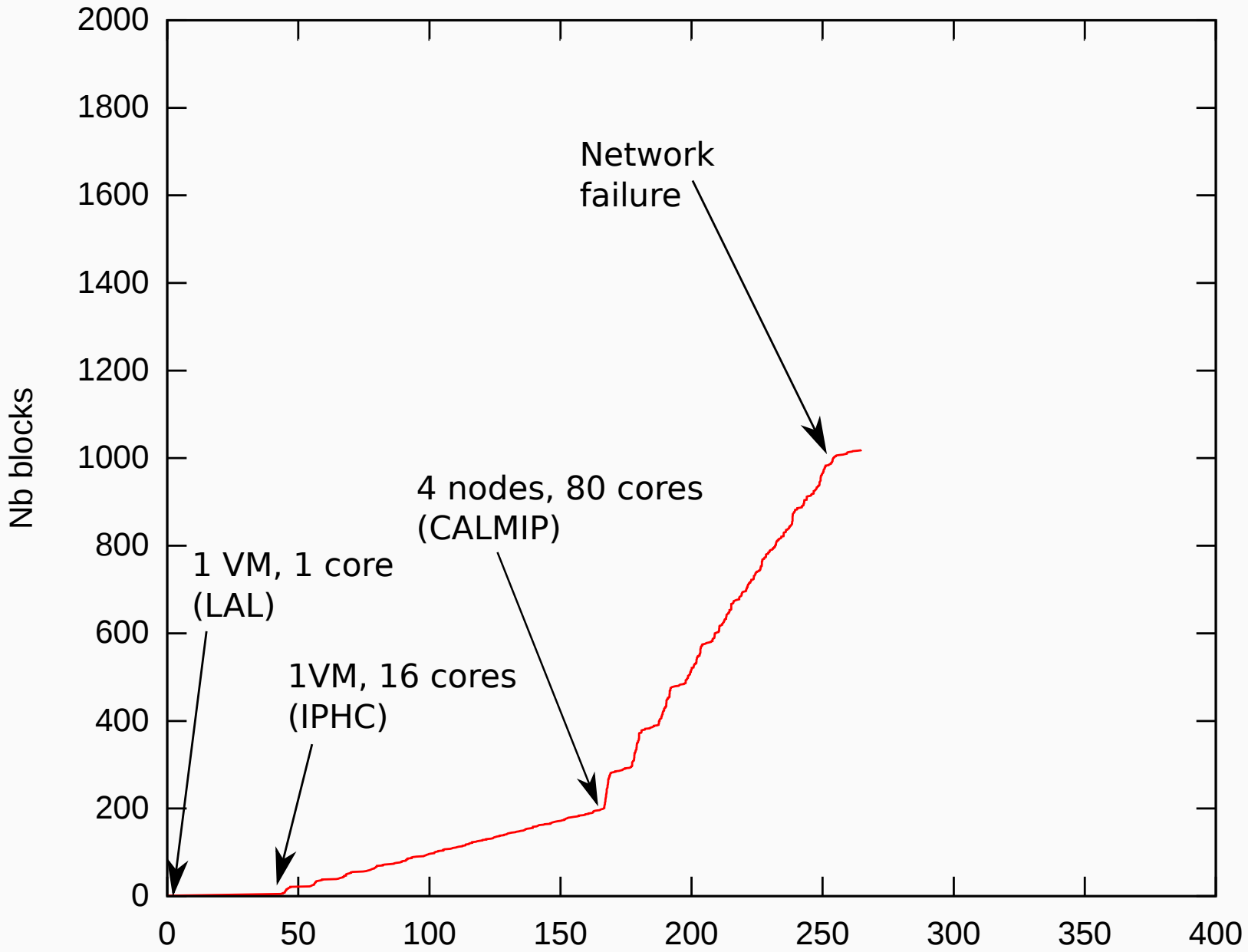


4. Simulation d'une coupure réseau

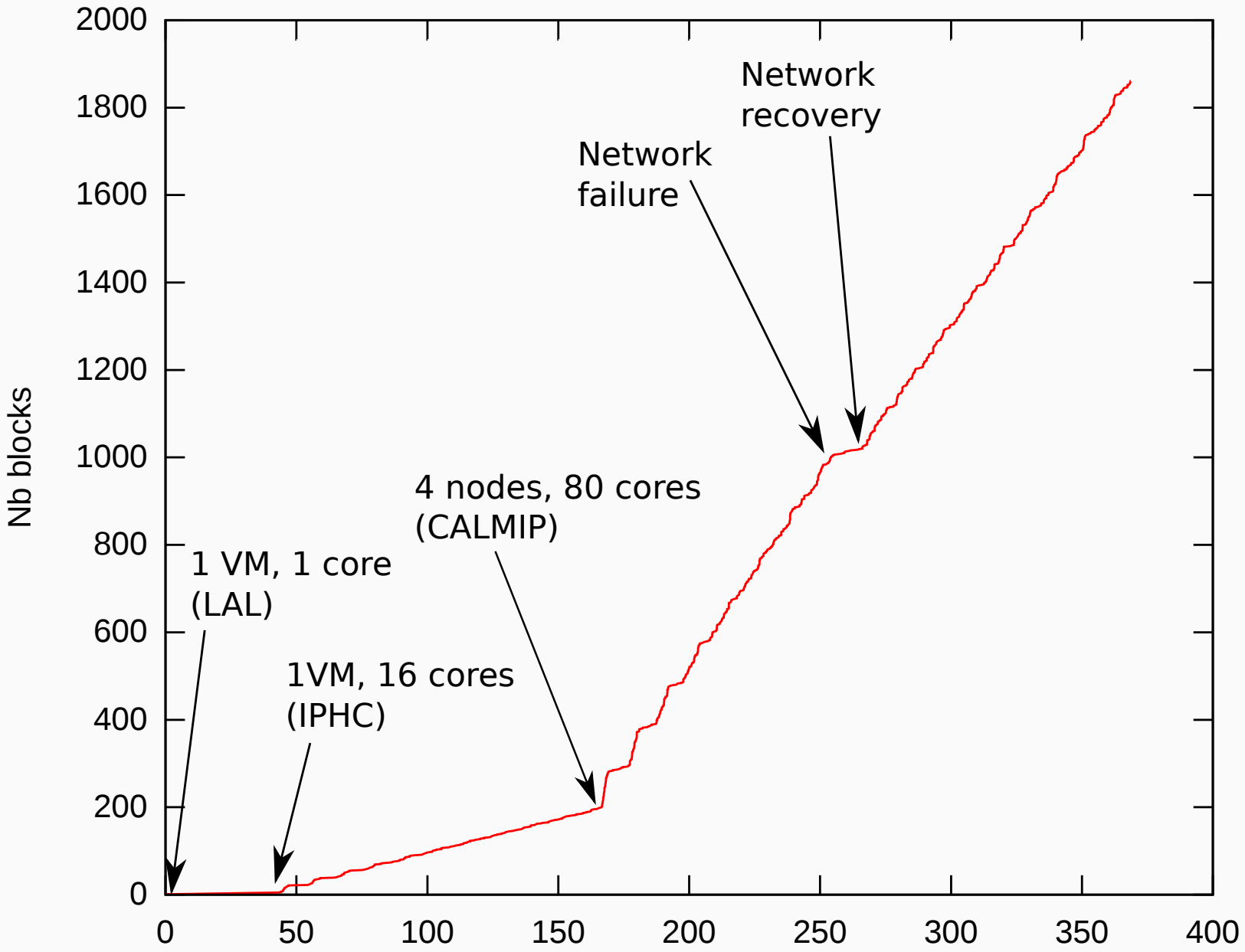


- On tue le proxy
- On attend 20 secondes
- On relance le proxy

4. Simulation d'une coupure réseau



4. Simulation d'une coupure réseau



Nombre de pas Monte Carlo par seconde par cœur:

	Pas MC/s/cœur
LAL	4152.9
IPHC	5715.5
CALMIP	5638.1

IPHC 1.4% plus efficace que CALMIP (turbo?)

RÉSUMÉ

RÉSUMÉ

- Notre code OpenMP (Quantum Package) fonctionne très bien sur des VMs
- La virtualisation n'est pas visible si on a accès directement au CPU
- Nous avons pu lancer un calcul distribué entre deux Clouds et un meso-centre avec succès

Perspectives

- Automatiser le déploiement des simulations (proxy, etc)
- Ajouter des interfaces web pour faciliter l'utilisation
- Utiliser le cloud pour la diffusion des codes (démos)
- Étudier la consommation énergétique
- Regarder si on peut utiliser QMC=Chem comme unikernel (MirageOS)

REMERCIEMENTS

- France Grilles (LAL, IPHC, IRIT, LUPM, CC-IN2P3)
- Cécile Cavet (LAL)
- Jérôme Pansanel (IPHC)
- François Thiebolt (IRIT)
- Nicolas Renon, Pierrette Barbaresco (CALMIP)