CNRS - INPT - UPS - UT1 - UT2J

Gestion d'énergie dans les systèmes distribués

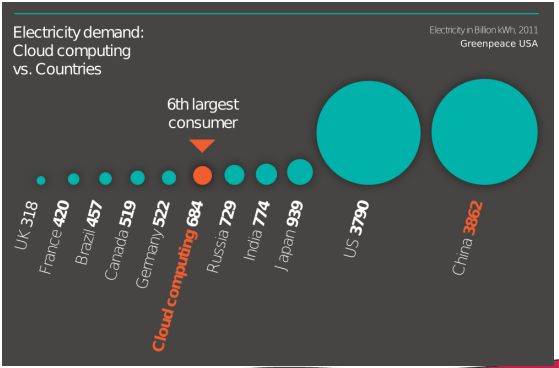
Georges Da Costa

École Jeunes Chercheurs sur
l'Efficacite Énergétique des Réseaux et Systèmes Distribués

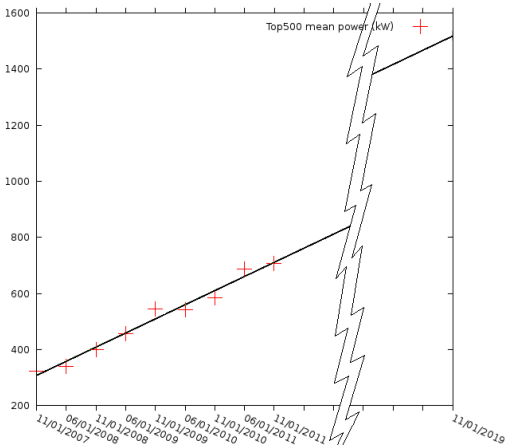
24 mai 2016

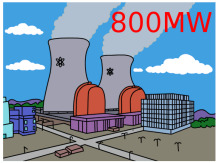
Impact des centres de calculs cloud

- Centre de calcul *récent* 40000 serveurs, 500000 services (machines virtuelles). Google, Facebook > 1million serveurs
- Les centres de calculs consomment beaucoup
  - 2000 : 70 TWh
  - 2007 : 330 TWh, 2% production mondiale de CO_2
  - 2011 : 6eme pays consommateur d'électricité au monde
  - 2020 : 1000 TWh
- En augmentation
  - 2014 : 8.5% des centres de calculs considérés comme suffisants au delà de 2015



Impact des centres de calculs





- China Telecom Inner Mongolia Information Park
  - 150 MW
- Tianhe-2
  - 17.8MW

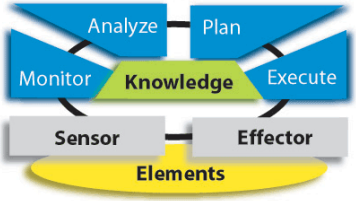
Des centres de calcul soutenables

- Actions à plusieurs niveaux orthogonaux
  - Matériel: changer les serveurs ou l'infrastructure de refroidissement
    - Si entropie constante, consommation théorique = 0
  - Applicatif: réécrire les applications en changeant de paradigme*ou de bibliothèque
  - Intergiciel: gère l'état des serveurs et des services/applications
- Intergiciel: coût minimal, impact maximal
  - OpenStack : 30% de part de marché en 2014
  - Solutions OpenSource : 43% (+72% en 2 ans)

* Georges et al. *Exascale machines require new programming paradigms and runtimes*, SFI journal, 2015

Principes de fonctionnement d'un intergiciel

- Deux buts :
  - Gérer (besoins, erreurs, fautes, surchauffes)
  - Optimiser (énergie, performance)
- Leviers/Effecteurs
  - Allumage, extinction, DVFS
  - Migration (x86/ARM)*, réduction de ressources, suspension
- Méthodes
  - Souvent dans la réalité : Humain ou règles
  - En recherche : Boucle autonome



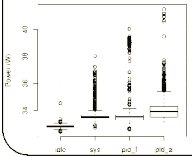
Boucle MAPE-K ©IBM

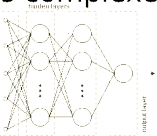
* Violaine et al., *Big, Medium, Little : Reaching Energy Proportionality with Heterogeneous Computing Scheduler*, Parallel Processing Letters, journal, 2015.

Boucle autonomique

Capteurs complexes

Impact des mesures





Programmation linéaire

Métriques

Contraintes :

$$\forall h \quad p_h \in \{0,1\}$$

$$\forall j \quad \sum_h e_{jh} = 1, \quad \forall j,h \quad e_{jh} \in \{0,1\}, o_{jh} \leq e_{jh}, m_{jh} \leq e_{jh}$$

$$\forall j \quad v_j \in \{0,1\}, \quad \forall h \quad \sum_j o_{jh} \leq v_j, \sum_j m_{jh} \leq v_j$$

$$\forall j,h \quad p_h \leq (1-e_{jh}) + v_j$$

$$\forall j,h \quad v_j \leq (1-e_{jh}) + p_h$$

$$\forall h \quad \sum_j o_{jh}/r_j \leq v_j$$

$$\forall j \quad \sum_h m_{jh} \leq v_j$$

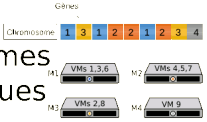
Réduire la puissance sous contrainte de performance :

$$\forall j \quad \sum_h o_{jh}/r_j > Seuil$$

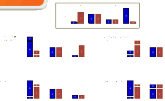
$$min(\sum_h (P_h^{min} + \sum_j o_{jh} \times (P_h^{max} - P_h^{min}))$$

Mesure → Analyse → Planifie → Execute → Mesure

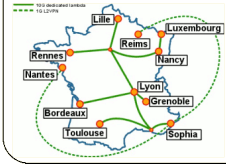
Algorithmes Génétiques



Vector packing



Leviers réels



Simulations

MELBOURNE CLOUDS LAB

Plan : Comment gérer efficacement un centre de calcul ?

- Efficacement ?
  - Il faut pouvoir comparer (**modèles & métriques**)
- Gérer veut dire décider
  - Outils de **mesure**
  - **Outils d'évaluation** : expérimentation, simulation
  - Approches exactes et heuristiques pour la **décision**
- L'**évolution** des centres de calcul
  - Fédération de centres de calcul
  - Optimisation multi-niveau

Plan

1 Boucle autonome
- Modèles et Métriques
- Mesures
- Outils d'évaluation

2 Décision
- Placement
- Fédération de clouds
- Data center in the box

3 Évolution, optimisation des noeuds
- Gros grain
- Moyen Grain
- Grain fin

4 Et au delà

Modéliser un système

Pour gérer un système il faut :

- Connaître toutes les actions possibles
- Savoir quelle est la(/les) plus adaptée

Ce qui se traduit par :

- Modéliser l'impact et la modalité (temps, énergie, ...) de ces actions
- Être capable de comparer deux scenarii

Impact des leviers : DVFS

Puissance dynamique délivrée par un composant CMOS :

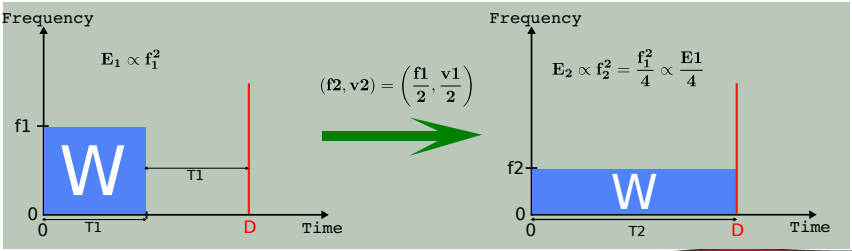
$P_{cmos} = C_{eff} \times V^2 \times f$

avec, C_{eff} la capacitance effective *, V le voltage et f la fréquence

* grandeur physique : capacité d'un composant à s'opposer au changement de voltage entre ses bornes

Énergie dépensée par tâche :

$$\underbrace{E = P * T \propto T * V^3 \text{ , avec } V \propto f \text{ et } T \propto 1/F\text{, alors } E \propto f^2}$$



Modélisation multi-échelles du monde réel

Plusieurs modèles décrivant la puissance consommée par un serveur :

- Classique : affine (erreur E∼10-15%)

$$Power = P_{min} + Load \times (P_{max} - P_{min})$$

Modélisation multi-échelles du monde réel

Plusieurs modèles décrivant la puissance consommée par un serveur :

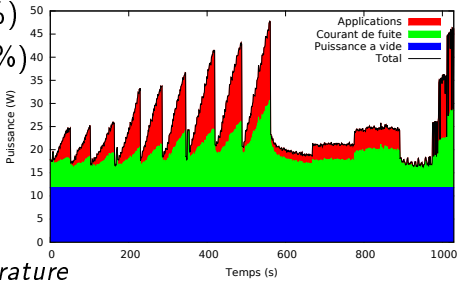
- Classique : affine (erreur E∼10-15%)
- Plus fin : Voltage/fréquence processeur (E∼5-9%)

$$Power = P_{min} + Load \times \alpha Voltage^2 Frequency$$

Modélisation multi-échelles du monde réel

Plusieurs modèles décrivant la puissance consommée par un serveur :

- Classique : affine (erreur E∼10-15%)
- Plus fin : Voltage/fréquence processeur (E∼5-9%)
- Encore plus fin: Température processeur (E∼4-7%)

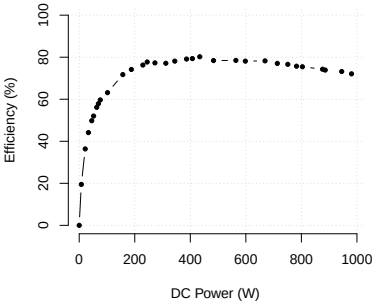


$$Power = P_{min} + Load \times \alpha Voltage^2 Frequency + \lambda Temperature$$

Modélisation multi-échelles du monde réel

Plusieurs modèles décrivant la puissance consommée par un serveur :

- Classique : affine (erreur E∼10-15%)
- Plus fin : Voltage/fréquence processeur (E∼5-9%)
- Encore plus fin: Température processeur (E∼4-7%)
- Ne pas oublier les biais : **alimentation électrique E∼2-3%**, refroidissement, ...

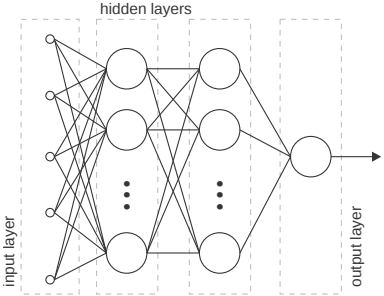


$$Power_{DC} = \omega_0 + \omega_1 Power_{AC} + \omega_2 Power_{AC}^3$$

Modélisation multi-échelles du monde réel

Plusieurs modèles décrivant la puissance consommée par un serveur :

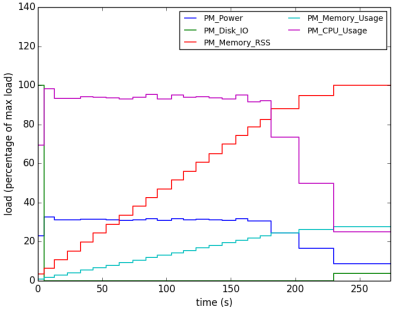
- Classique : affine (erreur E∼10-15%)
- Plus fin : Voltage/fréquence processeur (E∼5-9%)
- Encore plus fin: Température processeur (E∼4-7%)
- Ne pas oublier les biais : **alimentation électrique E∼2-3%**, refroidissement, ...
- Méthodes par apprentissage (réseaux de neurones, E∼2%) *



* Leandro et al., *Towards a generic power estimator*, CSRD journal, 2015

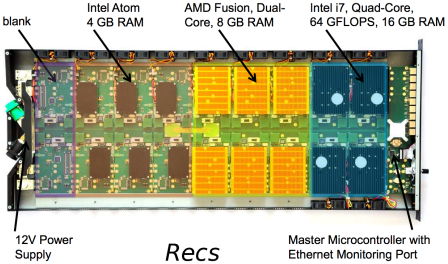
Modéliser un centre de calcul

- Un grand nombre de types d'éléments
  - Applications
    - Process: Traces, collection haut niveau puis abstraction



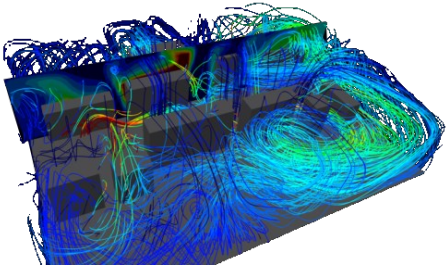
Modéliser un centre de calcul

- Un grand nombre de types d'éléments
  - Applications
    - Process: Traces, collection haut niveau puis abstraction
  - Serveurs



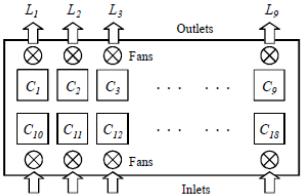
Modéliser un centre de calcul

- Un grand nombre de types d'éléments
  - Applications
    - Process: Traces, collection haut niveau puis abstraction
  - Serveurs
  - Infrastructure



Modéliser un centre de calcul

- Un grand nombre de types d'éléments
  - Applications
    - Process: Traces, collection haut niveau puis abstraction
  - Serveurs
  - Infrastructure
- Et leurs interactions
  - Thermiques (D-Matrix)*
  - Entre applications

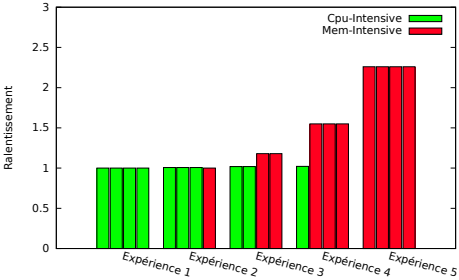


$$d_{x,k} = \begin{cases} 1, & \text{if } x = k \\ 0.84, & \text{if } x = k+9 \\ 0, & \text{otherwise} \end{cases}$$

* Hong Yang et al.,*Energy-efficient and thermal-aware resource management for heterogeneous datacenters*, SUSCOM journal, 2014.

Interactions simples d'applications

- Deux types d'applications mono-thread
  - Application 1 : Cpu-Intensive
    Limitée par le processeur
  - Application 2 : Mem-Intensive
    Limitée par la mémoire
- Exécution sur un quadri-cœur
  - Applications 1 : Indépendantes
  - Applications 2 : Impact croisé fort



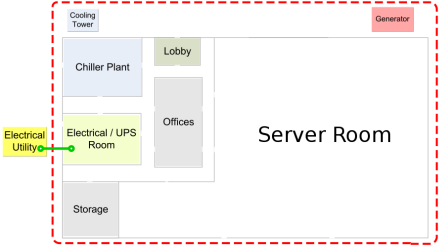
Métriques

- Valeurs directes :
  - Charge processeur et mémoire, puissance, Température, ...
  - Objectif : Ça marche bien ? Comparer deux centres, intergiciels, logiciels,...
- 40000 serveurs, 500000 services → Besoin de métriques simples
  - Consommation **et** performance
- Difficile à standardiser, surtout la performance
  - Dépend du service, de l'implémentation, ...
- Métrique classique : PUE

Georges et al. *Data Centres Sustainability Cluster Activities Task 3*. Rapport de recherche 3. European Commission, 2014

PUE : Power Usage Effectiveness

- Ratio électricité totale/électricité IT
- En moyenne 1.7 en 2014
- Standardisation initiée par GreenGrid
- Où s'arrête la partie IT
  - Alimentation ? Ventilateur sur la carte mère ? Processeur ?
- Ne sert que dans un cadre très précis



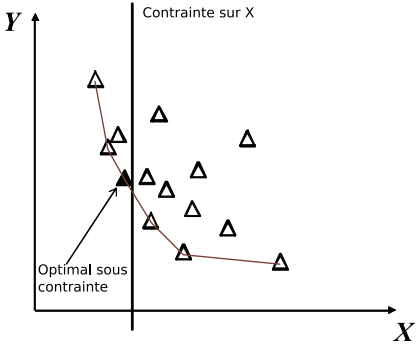
PUE : Power Usage Effectiveness

- Ratio électricité totale/électricité IT
- En moyenne 1.7 en 2014
- Standardisation initiée par GreenGrid
- Où s'arrête la partie IT
  - Alimentation ? Ventilateur sur la carte mère ? Processeur ?
- Ne sert que dans un cadre très précis

- Surcoût constant (100), partie IT 100 à 200 suivant la charge
- Pour un même service rendu deux logiciels
  1. Charge moyenne 75%
     PUE = 275/175 = 1.57
  2. Charge moyenne 100%
     PUE = 300/200 = 1.5

Problème du multi-objectif

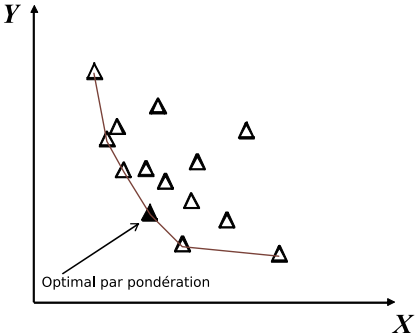
- Impossible à définir dans l'absolu, besoin d'un contexte, d'un but
- Formaliser les métriques simples* : Dynamisme, Énergie, Performance, Résilience
- Plusieurs méthodes classiques
  - Optimisation sous contrainte



* Tom et al., *Quality of Service Modeling for Green Scheduling in Clouds*, SUSCOM journal, 2014.

Problème du multi-objectif

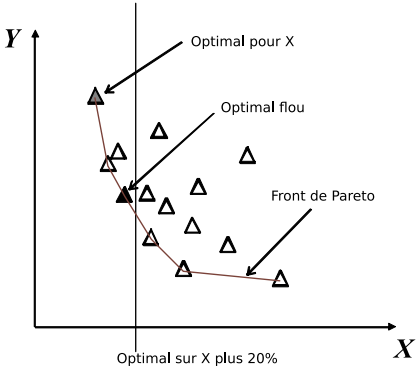
- Impossible à définir dans l'absolu, besoin d'un contexte, d'un but
- Formaliser les métriques simples* : Dynamisme, Énergie, Performance, Résilience
- Plusieurs méthodes classiques
  - Optimisation sous contrainte
  - Pondération d'objectif



* Tom et al., *Quality of Service Modeling for Green Scheduling in Clouds*, SUSCOM journal, 2014.

Problème du multi-objectif

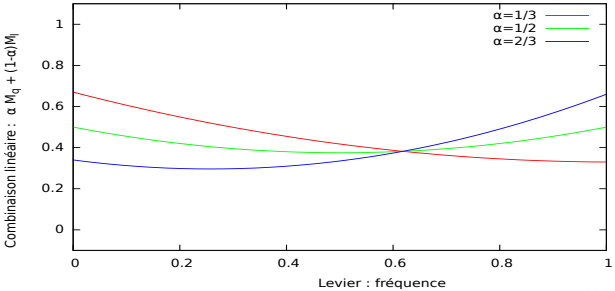
- Impossible à définir dans l'absolu, besoin d'un contexte, d'un but
- Formaliser les métriques simples* : Dynamisme, Énergie, Performance, Résilience
- Plusieurs méthodes classiques
  - Optimisation sous contrainte
  - Pondération d'objectif
  - Pondération floue† (contrainte par relâchement de l'optimal)



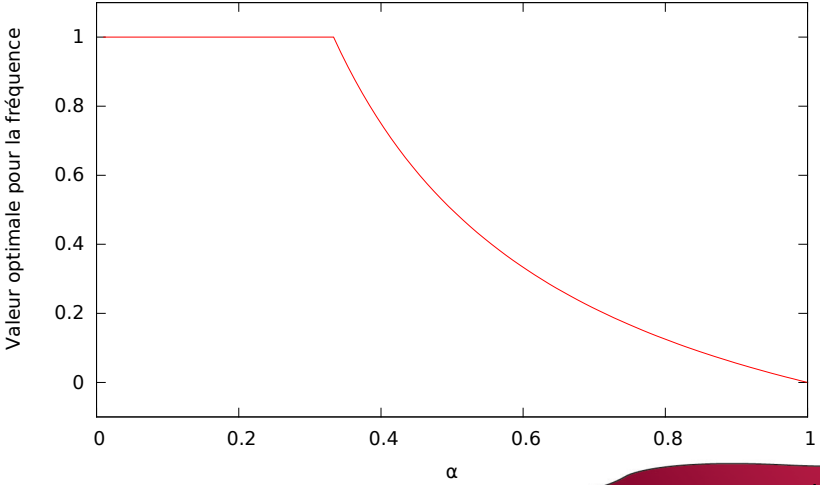
† Hong Yang et al. *Energy-efficient and thermal-aware resource management for heterogeneous datacenters*, SUSCOM journal, 2014.

Exemple: Performance et puissance

- Deux métriques, une linéaire, une quadratique
  - M_q est la partie puissance, donc quadratique en la fréquence (x)
  - M_l est la partie performance, donc linéaire en la fréquence (x)
- $Obj_\alpha(x) = \alpha M_q + (1-\alpha)M_l = \alpha x^2 + (1-\alpha)(1-x)$
- α est le coefficient de pondération



Comment choisir α ?



Suite de tests

- Deux grande catégories
  - Suites dédiées (services Web, base de données, HPC,...)
  - Suites génériques
- Scientifique, Gestionnaire d'infrastructure : Applications boîtes noires
- Le système doit être le meilleur dans tous les cas
- Suite de tests avec couverture maximale
  - Mêmes ressources/Puissances différentes
  - Ressources différentes/Puissances identiques

Georges et al., *Energy- and Heat-aware HPC Benchmarks*, EuroEcoDc workshop, 2013

Conclusion

Étude d'un système

- Pour quel usage ?
- Trois notions sont liées pour utiliser la réponse :
  - Équilibre du front de précision de la modélisation*
  - Objectifs utilisés pour choisir la méthode de comparaison
  - Scenarii pour réaliser la comparaison

* Georges et al., *Modèles fluides pour l'économie d'énergie dans les grilles par migration : une première approche*, RenPar conférence, 2009

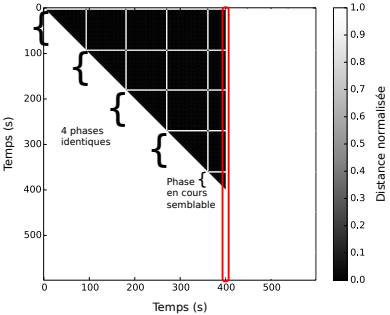
Infrastructure de mesure

- Base pour pour prendre la décision
- Base pour évaluer les métriques
  - Infrastructure classique (*nagios*, *ganglia*, ...)
  - Problème du passage à l'échelle
    - Des valeurs inutilisées ou agrégées tardivement
  - Des mesures (processeur, mémoire), pas de la connaissance
    - Besoin de mesures haut niveau
    - Une (phase d') application
    - La puissance électrique consommée par une application

Quelle (phase d') application s'exécute

- Une phase : comportement localement régulier
  - Équivaut à consommation de ressource constante
  - Mesures systèmes constantes
- Détection puis identification
  - Signature d'une phase
  - Phase équivalente $\sim$ impact équivalent

Landry et al., *Exploiting performance counters to predict and improve energy performance of HPC systems*, FGCS journal, 2014.



Matrice de similitude des mesures systèmes (WRF : Weather Research and Forecasting)

Détection externe d'application

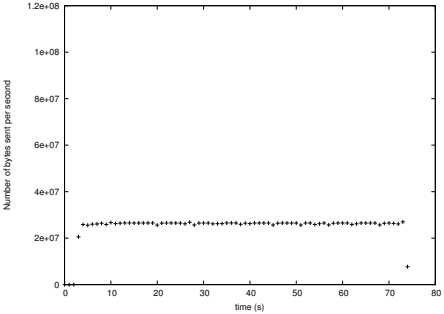
- Récupérer les valeurs systèmes est intrusif
- Diminuer le nombre de valeurs
- Récupérer des valeurs externes (puissance, réseau)
- S'autoriser les outils statistiques
- Étudier le comportement au cours du temps

Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011

Détection externe d'application

- Récupérer les valeurs systèmes est intrusif
- Diminuer le nombre de valeurs
- Récupérer des valeurs externes (puissance, réseau)
- S'autoriser les outils statistiques
- Étudier le comportement au cours du temps

Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011

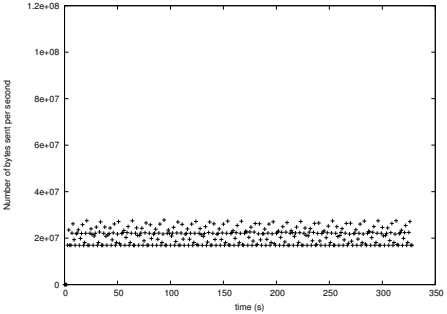


benchmark CG (NPB)

Détection externe d'application

- Récupérer les valeurs systèmes est intrusif
- Diminuer le nombre de valeurs
- Récupérer des valeurs externes (puissance, réseau)
- S'autoriser les outils statistiques
- Étudier le comportement au cours du temps

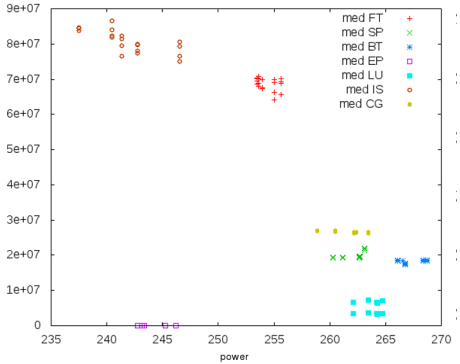
Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011



benchmark SP (NPB)

Détection externe d'application

- Récupérer les valeurs systèmes est intrusif
- Diminuer le nombre de valeurs
- Récupérer des valeurs externes (puissance, réseau)
- S'autoriser les outils statistiques
- Étudier le comportement au cours du temps



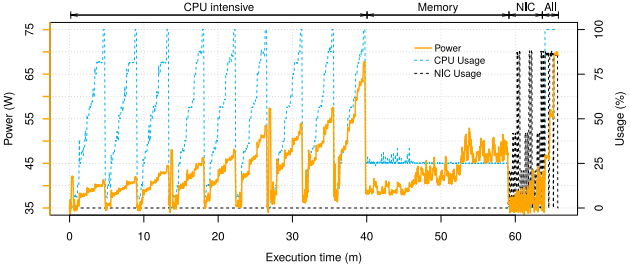
Georges et al., *Characterizing applications from power consumption : A case study for HPC benchmarks*, ICT-GLOW Symposium, 2011

Puissance des serveurs et applications

- Wattmètres pas toujours disponibles (niveau application)
- Modèle liant mesures systèmes à la puissance
- Analytique
  - Utilise les Datasheets. Très simple à mettre en oeuvre : PowerAPI

Puissance des serveurs et applications

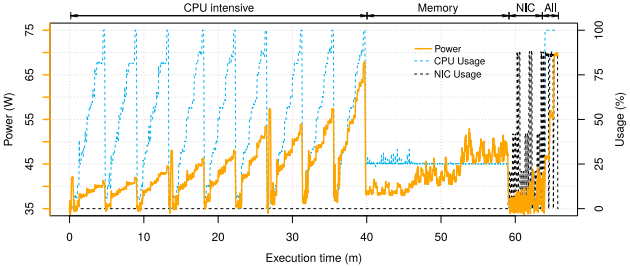
- Wattmètres pas toujours disponibles (niveau application)
- Modèle liant mesures systèmes à la puissance
- Méthode par apprentissage
  - Bonne couverture de la base d'apprentissage et faible impact de la mesure



Charge synthétique générique, 220 valeurs mesurées (4% augmentation puissance), 8 gardées

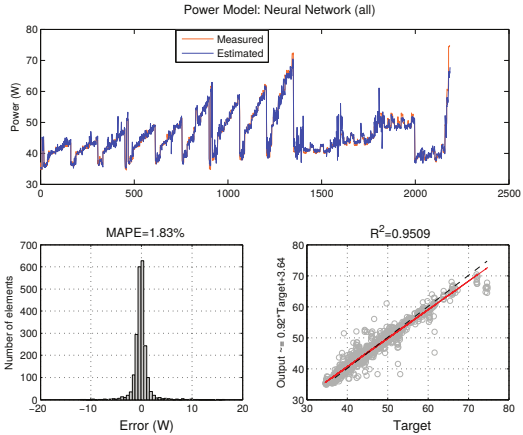
Puissance des serveurs et applications

- Wattmètres pas toujours disponibles (niveau application)
- Modèle liant mesures systèmes à la puissance
- Méthode par apprentissage
  - Bonne couverture de la base d'apprentissage et faible impact de la mesure

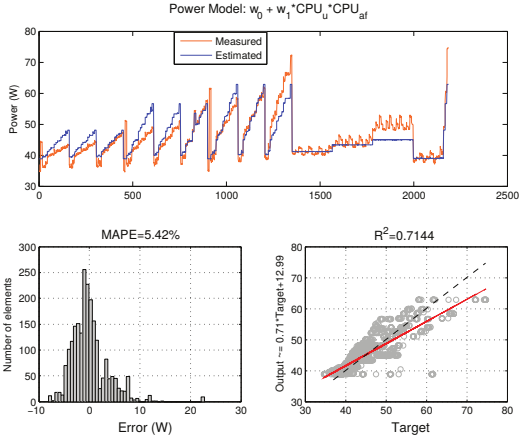


20% des valeurs prises par des jeux de tests HPC ne sont pas atteintes

Réseau de neurones VS modèle capacitif

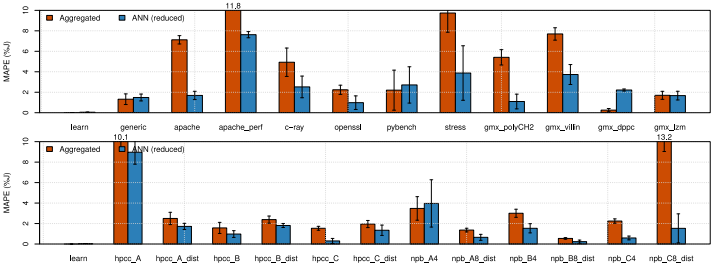


(a) ANN regression using all KPIs.



(b) Calibrated capacitive model.

Résultat des méthodes par apprentissage



Erreur de deux modèles : **Aggregated** (régression linéaire sur somme des sous-modèles) et **ANN** (réseau de neurones)

- Majorité de modèles 5% d'erreur
  - Référence du domaine : Rivoire et Al. *A Comparison of High-Level Full-System Power Models*

Leandro *Modeling the power consumption of computing systems and applications through Machine Learning techniques*, 2015

Conclusion

- Approches actuelles nécessitent une vue globale
  - Passage à l'échelle
  - Problème de latence
  - Beaucoup de décisions sont locales (DVFS, échange de charge,...)
- Problèmes ouverts
  - Granularité de la mesure adaptative et multi-échelle
  - Indépendance temporelle et spatiale
  - Génération suite synthétique avec couverture maximale

Outils d'évaluation: expérimentation, simulation

- Pour améliorer il faut comparer
- Trois grandes méthodes
  - Modèles mathématiques
  - Simulation
  - Expérimentation

Programmation linéaire

- Expliciter toutes les contraintes par des équations linéaires

Exemple : Un service est sur un serveur unique

- Soit e_{jh} la présence du service j sur le serveur h
- $e_{jh} = 1$ ssi le service j est sur le serveur h
- $\forall j, h \quad e_{jh} \in \{0, 1\}$,
  $\forall j \quad \sum_h e_{jh} = 1$

Damien et al., *Energy-Aware Service Allocation*, FGCS journal, 2011

Programmation linéaire

- Expliciter toutes les contraintes par des équations linéaires
- Exprimer l'objectif sous forme d'une fonction à minimiser

Exemple : Minimiser la puissance totale consommée

- P_h^{stat} et P_h^{dyn} : puissances statique et dynamique du serveur h (modèle affine)
- Soit α_{jh} la fraction de ressources processeur du service j sur le serveur h
- $min \sum_h (P_h^{stat} + \sum_j \alpha_{jh} P_h^{dyn})$

Damien et al., *Energy-Aware Service Allocation*, FGCS journal, 2011

Programmation linéaire

- Expliciter toutes les contraintes par des équations linéaires
- Exprimer l'objectif sous forme d'une fonction à minimiser
- Formalise les leviers et leur impact
- Approximation du monde réel (phénomènes quadratiques)
- Résolution exacte possible pour les petits cas

Contraintes :

$$\forall h \quad p_h \in \{0,1\}$$

$$\forall j \quad \sum_h e_{jh} = 1, \quad \forall j,h \quad e_{jh} \in \{0,1\}, \alpha_{jh} \leq e_{jh}, m_{jh} \leq e_{jh}$$

$$\forall j \quad v_j \in \{0,1\}, \quad \forall h \quad \sum_j \alpha_{jh} \leq v_j, \sum_j m_{jh} \leq v_j$$

$$\forall j,h \quad p_h \leq (1 - e_{jh}) + v_j$$

$$\forall j,h \quad v_j \leq (1 - e_{jh}) + p_h$$

$$\forall h \quad \sum_j \alpha_{jh}/r_j \leq v_j$$

$$\forall j \quad \sum_h m_{jh} \leq v_j$$

Réduire la puissance sous contrainte de performance :

$$\forall j \quad \sum_h \alpha_{jh}/r_j > Seuil$$

$$min(\sum_h (P_h^{min} + \sum_j \alpha_{jh} \times (P_h^{max} - P_h^{min}))$$

Damien et al., *Energy-Aware Service Allocation*, FGCS journal, 2011

Passage à l'échelle (programmation linéaire)

- Méthode exacte: Temps de résolution exponentiel en fonction des variables entières
  - 6 serveurs, 16 services : 3 minutes (GLPK)

Passage à l'échelle (programmation linéaire)

- Méthode exacte: Temps de résolution exponentiel en fonction des variables entières
  - 6 serveurs, 16 services : 3 minutes (GLPK)
- Méthode des variables constantes
  - Fixe certaines variables, résout, change de variables, itère
  - Moins bon que l'optimal, mais peut être très rapide

Passage à l'échelle (programmation linéaire)

- Méthode exacte: Temps de résolution exponentiel en fonction des variables entières
  - 6 serveurs, 16 services : 3 minutes (GLPK)
- Méthode des variables constantes
  - Fixe certaines variables, résout, change de variables, itère
  - Moins bon que l'optimal, mais peut être très rapide
- Relaxation de la contrainte entière
  - Meilleur que l'optimal (un service à moitié sur une machine et à moitié sur une autre)

Passage à l'échelle (programmation linéaire)

- Méthode exacte: Temps de résolution exponentiel en fonction des variables entières
  - 6 serveurs, 16 services : 3 minutes (GLPK)
- Méthode des variables constantes
  - Fixe certaines variables, résout, change de variables, itère
  - Moins bon que l'optimal, mais peut être très rapide
- Relaxation de la contrainte entière
  - Meilleur que l'optimal (un service à moitié sur une machine et à moitié sur une autre)
- Avec les deux on peut donner un intervalle

Simulation

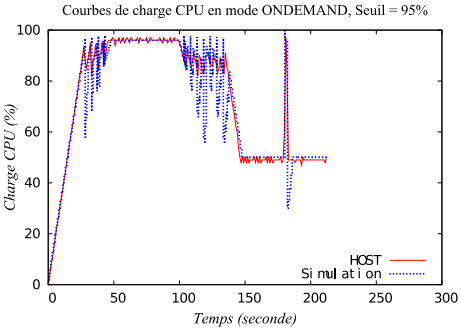
- Grand nombre de simulateurs existent : SimGrid, DCWorms, CloudSim, ...
- Besoins dans notre domaine
  - Modèle de cloud (migration, sur-allocation de ressources, fédération†)
  - Gestion du DVFS
  - Puissance électrique
  - Température
- La situation évolue
  - DVFS et gestion fine du cloud dans CloudSim
  - Thermique dans DCWorms*
  - DVFS et énergie dans SimGrid

* Wojtek et al., *Energy and thermal models for simulation of workload and resource management in computing systems*, SMPT journal, 2015. †Thiam et al., *Cooperative Scheduling Anti-load balancing Algorithm for Cloud*, CCTS workshop, 2013

Ajout du DVFS dans CloudSim

- Simulateurs du monde des Grilles de calcul
  - Grande stabilité dans le temps
  - Ressources utilisées à 100%
- DVFS provoque décalage d'événements
- Gestion temporelle fine (1/10 s)

Tom et al., *Energy-aware simulation with DVFS*, SMPT journal, 2013



Des simulateurs ad-hoc : Réinventer la roue ?

- Des simulateurs simplistes
- Permet de tester une idée à moindre coût
- Nécessaire de s'arrêter au bon niveau de complexité
  - Simulateur d'architectures hétérogènes*
  - Pas de simulation du réseau
  - Exemple: prouver l'utilité de l'hétérogénéité pour atteindre un système proportionnel

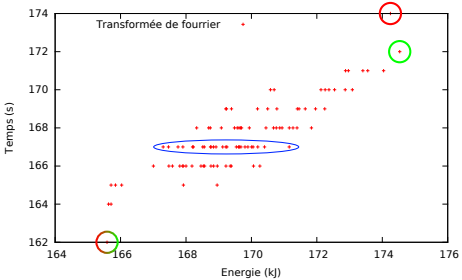
* Georges, *Heterogeneity: The Key to Achieve Power-Proportional Computing*, CCGrid conférence, 2013.

Expérimentation

- Un modèle est toujours approché
- Validation finale par expérimentation
- Compliqué par la nécessité d'avoir des informations électriques
  - A Lyon, parmi les premier à expérimenter la capture de puissance à grande échelle (GreenNet)
- Problème des mesures distribuées, conversions électriques, impact de la mesure (compteurs de performance)
- Problème de la reproductibilité

Reproductibilité d'une expérience

- Expérimentation *simple* de Transformée de Fourier (NPB)
- 100 expériences sur les mêmes machines (Grid'5000)
- Grande variation
  - Temps: 12s, 7% (Std. Dev. 3.2s)
  - Énergie: 9.3kJ, 5.5% (3kJ)
- Pour un même temps, 167s, différence de 4kJ
- Temps $\neq$ Énergie



Conclusion

- Tous les outils sont limités
  - Modélisation: Approximatif ou optimal inatteignable
  - Simulation: Approximatif
  - Expérimentation: Reproductibilité et très sensible
- Investissement très fort nécessaire
- Simulation : Bon rapport qualité/temps
- Difficulté de tester In-Vivo

Plan

1 Boucle autonome
- Modèles et Métriques
- Mesures
- Outils d'évaluation

2 Décision
- Placement
- Fédération de clouds
- Data center in the box

3 Évolution, optimisation des noeuds
- Gros grain
- Moyen Grain
- Grain fin

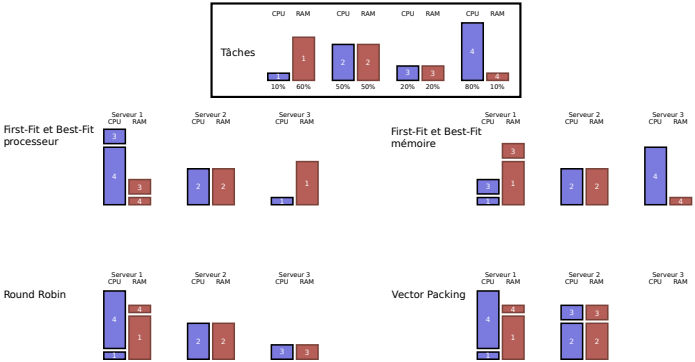
4 Et au delà

Approches exactes et heuristiques

- Deux problèmes
  - Placement
  - Temporalité
- Les heuristiques classiques du placement
  - Gloutons Best Fit, First Fit
  - Vector Packing (Glouton *gourmet*)
  - Algorithmes Génétiques

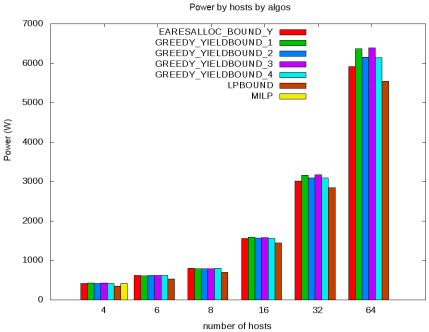
Les algorithmes gloutons classiques

- Caractéristiques
  - Mémoire
  - Processeur
- Tri des services
- Tri des machines
- Pas de remise en cause



Vector packing *gourmet*

- 4 objectifs dans la fonction de tri
  - Serveur est énergétiquement attractif
  - Rajouter la tâche ne surcharge pas
  - Serveur allumé
  - La tâche rétablit l'équilibre dans les ressources
- Temps "seulement"en $\mathcal{O}(J \times H \ln(H))$
- Mais la solution du *gourmet* est difficile à qualifier

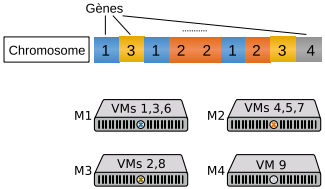


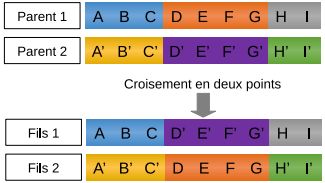
Damien et al., *Energy-Aware Service Allocation*, FGCS journal, 2012.

Algorithmes Génétiques

- Chromosome = Allocation
- Génération aléatoire initiale
- À chaque génération :
  - Croisements et mutation
  - Tri sur la métrique d'objectif
  - Garder les meilleurs

Tom et al., *Quality of Service Modeling for Green Scheduling in Clouds*, SUSCOM journal, 2014





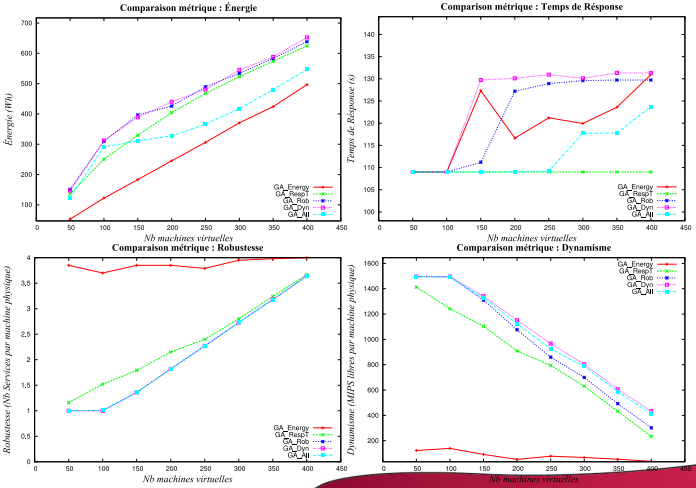
Métriques pour Algorithmes Génétique

- Contrairement aux gloutons on peut optimiser une métrique directement
- Exemples de métriques
  - Énergie, Performance, Résilience, Dynamisme

Nom GA	Coefficients appliqués aux métriques			
	Énergie	Temps de réponse	Robustesse	Dynamisme
GA_All	1	1	1	1
GA_Energy	1	0	0	0
GA_RespT	0	1	0	0
GA_Rob	0	0	1	0
GA_Dyn	0	0	0	1

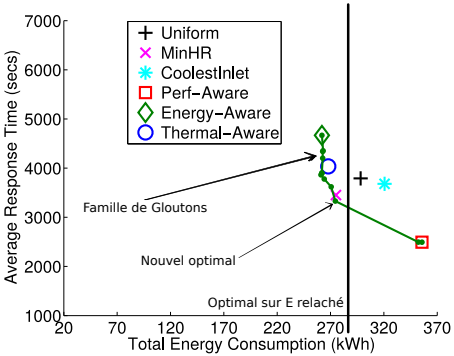
Résultats de l'algorithme génétique

- Chaque algorithme est meilleur dans son domaine (Énergie)
- GA_All très bon partout
- 400 services sur 110 machines, environ 40s
- Tenir compte d'une métrique est déjà très important



Glouton Flou

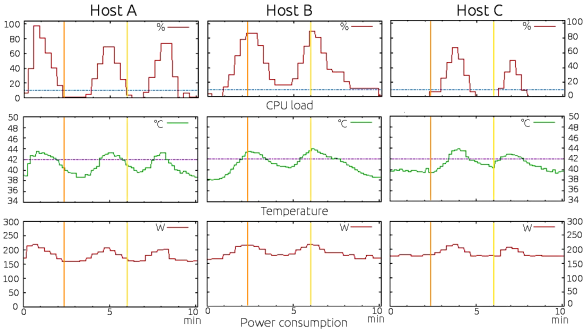
- Avantage du G.A. : viser un objectif
- Méthode similaire pour glouton
  - Famille de glouton
  - Garde le meilleur
  - Définir le *meilleur* ?
- Multi-objectif : flou



Hong Yang et al., *Multi-Objective Scheduling for Heterogeneous Server Systems with Machine Placement*, CCGRID conférence, 2014

Ce qui rend la puissance/énergie unique

- Le coté temporel
  - Inertie liée à la température
  - Allumage et extinction de serveurs
    - Sur- ou Sous-réservation
  - Les cycles sont parfois bon
- Les non linéarités
  - Équation de puissance
- Les boucles de rétroaction
  - Système de refroidissement

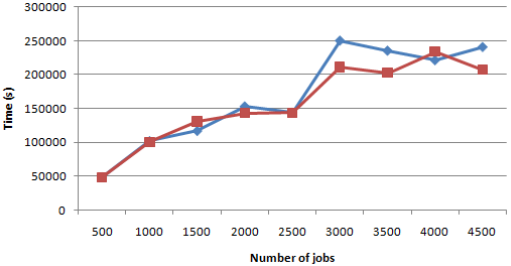


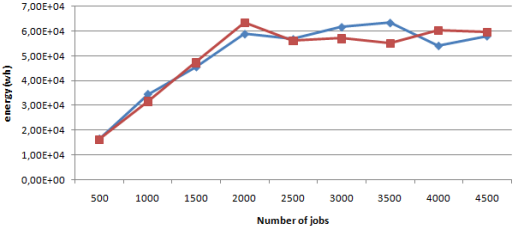
Violaine et al., *Thermal-aware cloud middleware to reduce cooling needs*, WETICE workshop, 2014

À l'échelle de la fédération de clouds

- Opérateur loue les ressources de ses concurrents (itinérance télécom)
- Méthode semblable au *super-peers*
- Distribué ou centralisé, mêmes performances

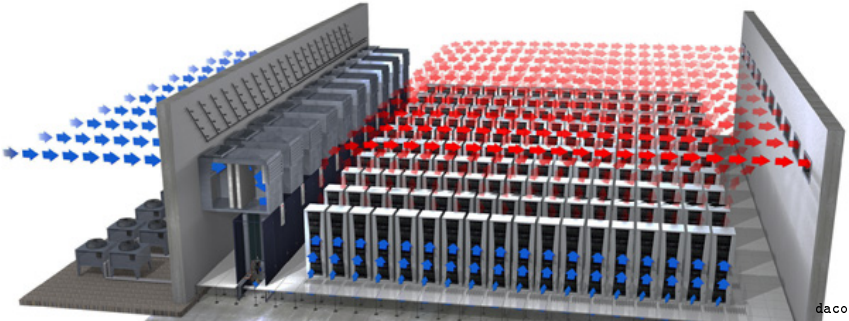
Thiam et al., *Cooperative Scheduling Anti-load balancing Algorithm for Cloud* : *CSAAC*, CCTS workshop, 2013





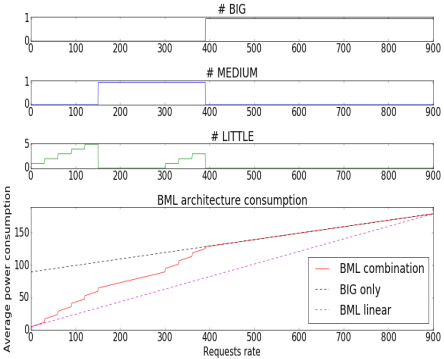
Follow the sun, Follow the moon

- Approches classiques
  - Consolidation entre les centres de calculs
  - Gestion coordonnée de la QoS (ex: CDN)
- Suivre les conditions des centres de calculs
  - La nuit, moins de frais de cooling
  - Le jour, plus de production renouvelable



À l'échelle du *Data center in the box*

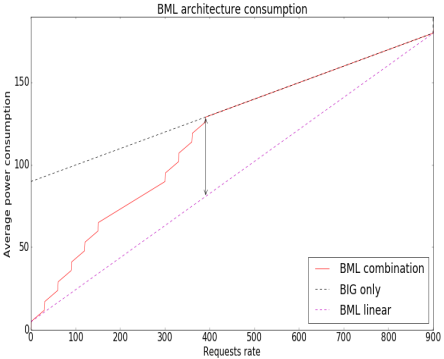
- Échelle du Rack
- Peu de services : Forte variabilité
- Adaptation parfaite à la charge
- Actuellement coûteux : Sur-coût initial
  - Climatisation
  - Tout ce qui est négatif dans le PUE
- Architecture *proportionnelle*



Violaine et al., *Big, Medium, Little : Reaching Energy Proportionality with Heterogeneous Computing Scheduler*, Parallel Processing Letters, journal, 2015.

À l'échelle du *Data center in the box*

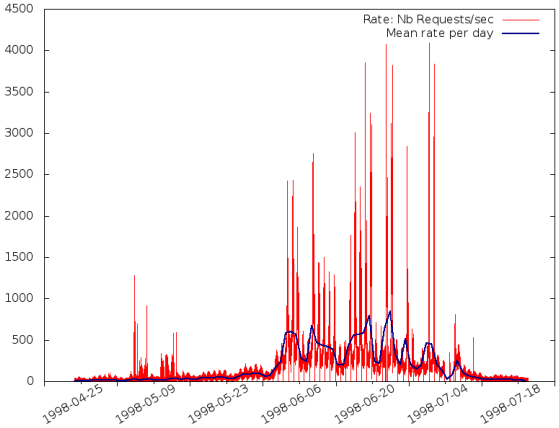
- Échelle du Rack
- Peu de services : Forte variabilité
- Adaptation parfaite à la charge
- Actuellement coûteux : Sur-coût initial
  - Climatisation
  - Tout ce qui est négatif dans le PUE
- Architecture *proportionnelle*



Violaine et al., *Big, Medium, Little : Reaching Energy Proportionality with Heterogeneous Computing Scheduler*, Parallel Processing Letters, journal, 2015.

Exemple de proportionnalité

- Accès au serveurs web de la coupe du monde de football 1998



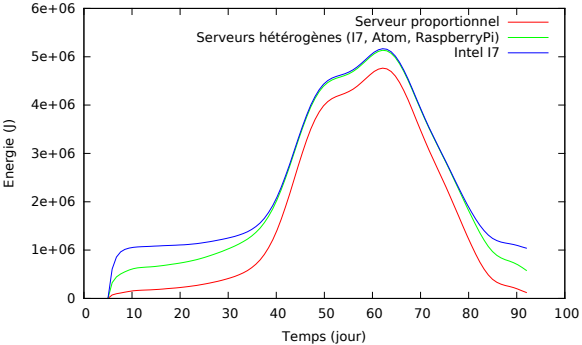
Architecture proportionnelle

- Application : serveur web
- Architecture matérielle
  - Raspberry Pi, Intel Atom et I7

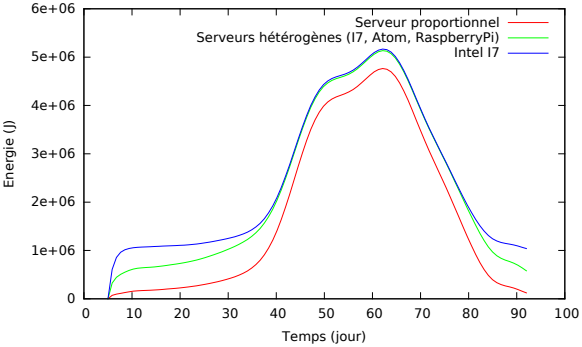
Processor	Watt range	Max request/s	Efficiency
Intel I7	11 - 42	353	.12 W/r/s
Intel Atom	8 - 9	34	.26 W/r/s
Raspberry Pi	2.56 - 2.81	5.6	.50 W/r/s

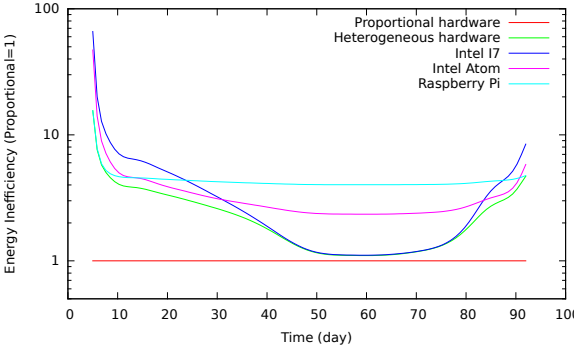
Da Costa, *Heterogeneity, the key to reach proportionality, CCGRID 2014*

Comparaison des cocktails d'architectures



Comparaison des cocktails d'architectures





Plan

1 Boucle autonome
- Modèles et Métriques
- Mesures
- Outils d'évaluation

2 Décision
- Placement
- Fédération de clouds
- Data center in the box

3 Évolution, optimisation des noeuds
- Gros grain
- Moyen Grain
- Grain fin

4 Et au delà

Vers le futur

- Un plus grand nombre de centres de calcul
  - Beaucoup de plus petits (gestion hybride)
    - La connaissance : ressource critique
  - Un grand nombre de tailles diverses
  - Quelques plus grands (2016 : 6 300 000 m^2)
- De manière globale des centres de calcul plus intégrés dans leur environnement
  - Aspects électriques
  - Aspects thermiques

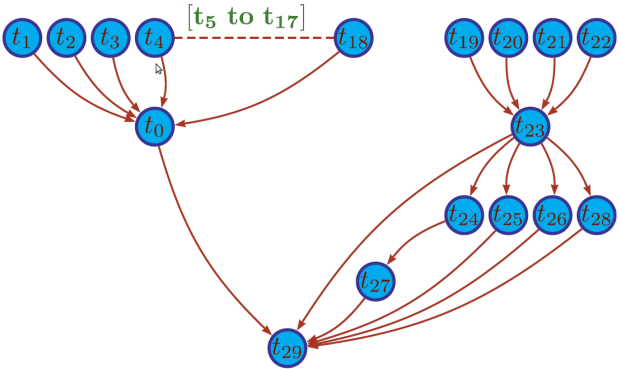
À l'échelle du nœud

- Trois temporalités
  - Gros grain (minute) : Fréquence optimale en fonction du graphe de tâches*
    - 13% d'économie d'énergie
  - Moyen Grain (seconde) : Détection de phase†
    - 20% d'économie d'énergie, 3% augmentation du temps
  - Grain fin (1/10s) : Politique noyau de gestion de fréquence‡
    - 25% d'économie d'énergie, 1% réduction du temps
- Pas de coordination entre les trois temporalités, pas d'objectif

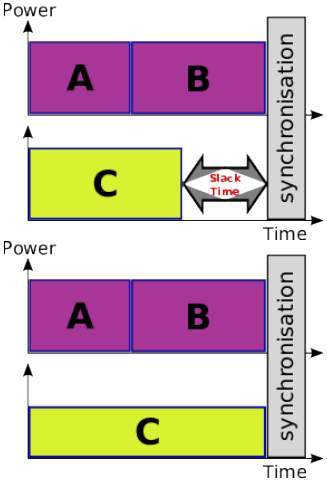
* Tom et al., *Energy-aware simulation with DVFS*, SMPT journal, 2013 †Landry et al., *Exploiting performance counters to predict and improve energy performance of HPC systems*, SUSCOM journal, 2014 ‡Georges et al., *DVFS governor for HPC: Higher, Faster, Greener*, PDP conférence, 2015

À l'échelle du nœud: Gros grain

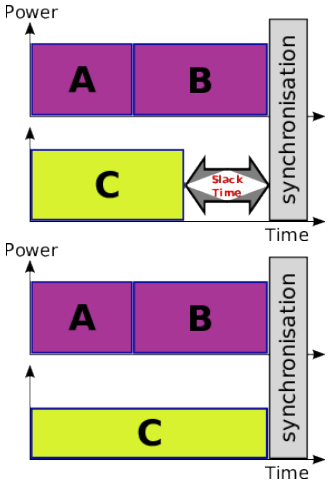
- Utilisation d'informations extérieures contextuelles
- Exemple au niveau du scheduler : DAG de tâches



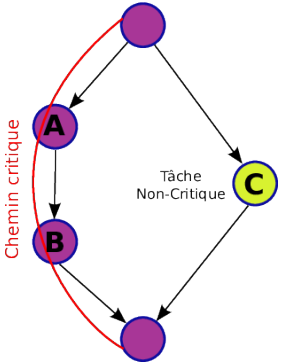
Coordination des vitesses des noeuds



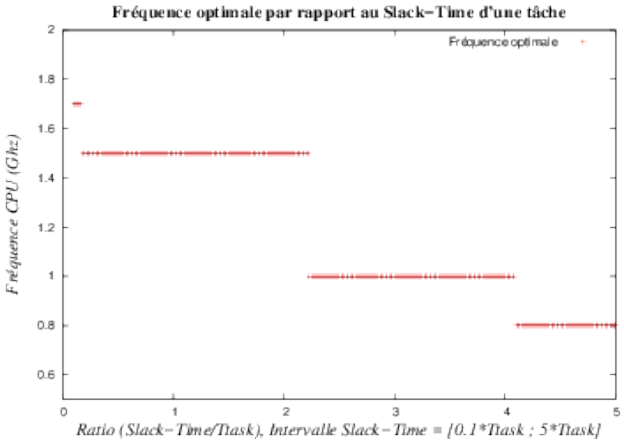
Coordination des vitesses des noeuds



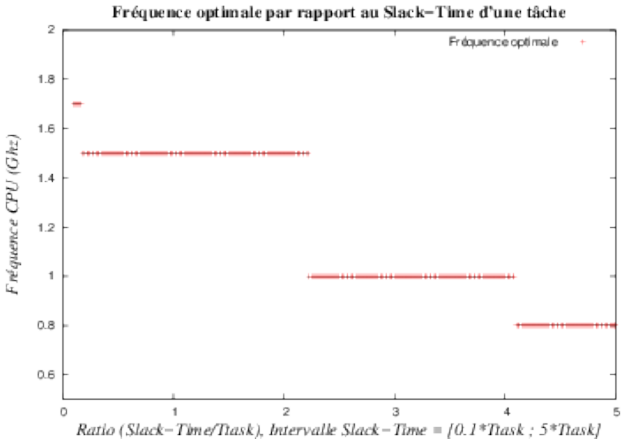
- Généralisation au chemin critique



Action au niveau du noeud



Action au niveau du noeud

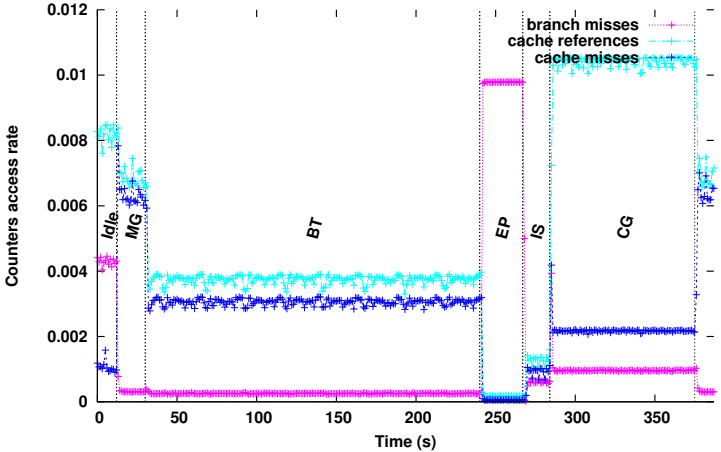


- Pour aller plus loin
  - Allumer/Éteindre les serveurs
  - Gérer la température

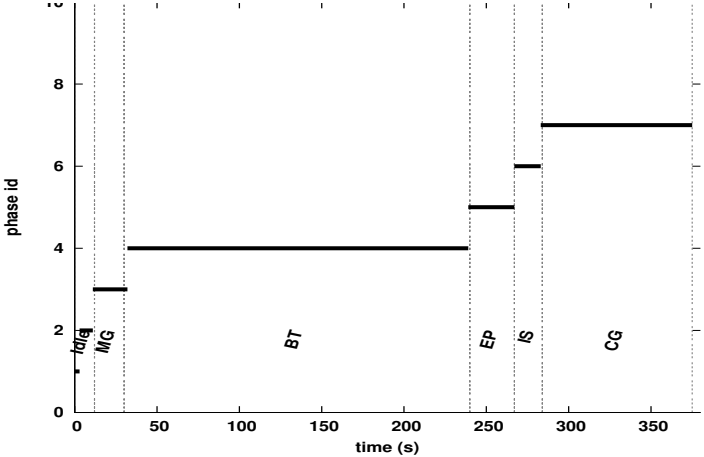
À l'échelle du nœud: Moyen grain

- Réactions à latence moyenne au niveau du noeud
  - Changement de fréquence processeur
  - Changement de mode du disque dur
  - Reconfiguration de la carte réseau
- Détection de la phase actuelle
- Réaction en fonction du profil
- Impact léger sur l'infrastructure

Consommation de ressources d'une application complexe



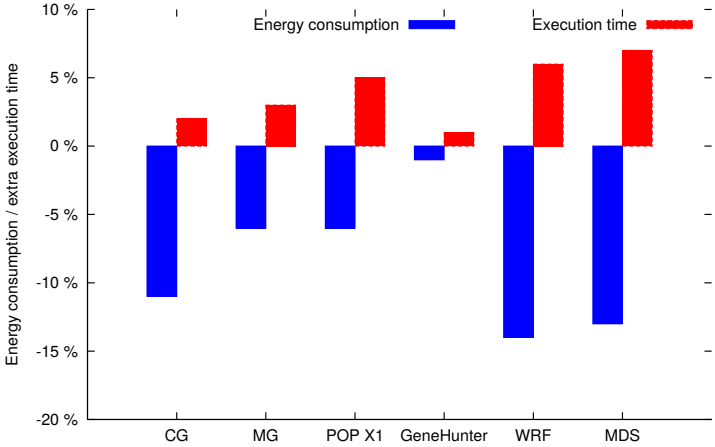
Les phases de consommation de ressources constantes



Méthode de décision

Label de phase	Décision de reconfiguration possible
compute-intensive	éteindre les bancs mémoires inutilisés; mettre le disque en sommeil; mettre le processeur à fréquence maximale; mettre les interfaces réseaux en sommeil.
memory-intensive	réduire la fréquence du processeur; mettre le disque en sommeil; ou réduire sa vitesse; allumer tous les bancs mémoires.
mixed	allumer tous les bancs mémoire; augmenter la fréquence du processeur; mettre le disque en sommeil; mettre les interfaces réseaux en sommeil.
communication intensive	éteindre les bancs mémoire; réduire la fréquence du processeur; allumer le disque.
IO-intensive	allumer les bancs mémoire; réduire la fréquence du processeur; augmenter la vitesse du disque.

Energie et performance, 28 noeuds



Grain fin = DVFS ?

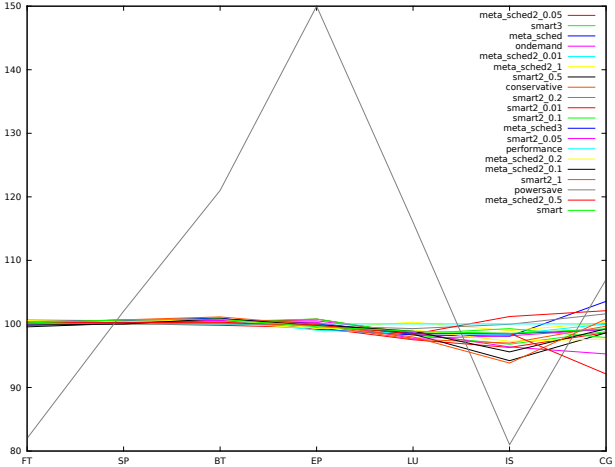
Valeurs relatives entre les gouverneurs performance et ondemand

Benchmark	FT	SP	BT	EP	LU	IS	CG
Time increase (%)	0	-3	-1	1	-2	2	0
Energy increase (%)	0	-3	-1	-1	-2	-1	-1

- Les applications HPC sont rarement en Idle... Surprise !
- Les bibliothèques MPI font de l'attente active

Classical HPC benchmarks from NPB (Nas Parallel Benchmark)

Le DVFS = fonction de la charge



Le DVFS a pourtant du potentiel

Valeurs relatives entre performance et powersave

Benchmark	FT	SP	BT	EP	LU	IS	CG
Time increase (%)	36	69	110	159	96	35	83
Energy increase (%)	-18	2	21	50	16	-19	7

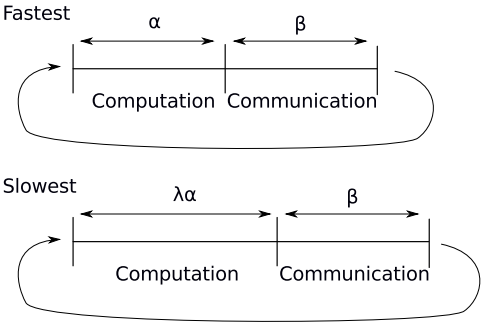
- Le temps augmente, mais jusqu'à 19% de réduction de la consommation d'énergie !

Hypothèse HPC

- États des applications
  - Calcul
  - Communications
  - Disk I/O
  - Idle

Hypothèse HPC

- États des applications
  - Calcul
  - Communications



Décision

- Énergie pour la fréquence max

$$(\alpha + \beta)P_1$$

- Énergie pour la fréquence min

$$(\lambda\alpha + \beta)P_2$$

On reste à fréquence maximum si on consomme moins d'énergie :

$$(\alpha + \beta)P_1 < (\lambda\alpha + \beta)P_2$$

Comment mesurer α et β

- Difficile de mesurer directement α et β
  - On veut du runtime, pas de l'instrumentation de code
- Facile de mesurer la bande passante réseau (avec B_m Bande passante maximale)

$$B_w = B_m \frac{\beta}{\alpha + \beta}$$

- En fait α et β ne sont pas important
  - On veut $\frac{\alpha}{\beta}$, i.e. le ratio entre le *time to compute* et le *time to communicate*

The great mix

Mélanger, servir:

$$B_w < \frac{B_m}{\lambda - 1}(\lambda - \frac{P_1}{P_2}) = B_1$$

B_1 : Bande passante limite à fréquence max pour utiliser ou non le DVFS

Dans l'autre sens

$$B_2 = \frac{B_m}{\lambda - 1}(\lambda \frac{P_2}{P_1} - 1)$$

B_2 : Bande passante limite à fréquence min pour utiliser ou non le DVFS

Ajout d'un hystéresis pour l'inertie

Algorithme NetSched

- Tous les 10emes de seconde, faire:
  - Si Current_Frequency = Slowest frequency et IBR $\leq .9B_1$
    - Changer la fréquence vers Fastest
  - Si Current_Frequency = Fastest frequency et IBR $\geq 1.1B_2$
    - Changer la fréquence vers Slowest
  - Sinon, ne rien faire

IBR : Incoming Byte Rate

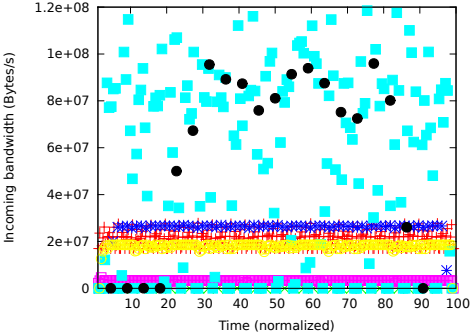
Environment expérimental

- Serveurs (thanks Grid5000)
  - Processeurs : bi Dual-Core AMD Opteron (2218)
  - Mémoire : 8GB
  - Carte réseau : Gigabyte Ethernet
  - Fréquence : 2.6GHz and 1GHz
  - Puissance électrique : $P_1 = 280W$ et $P_2 = 152W$
- Benchmark
  - 7 Nas Parallel Benchmark (NPB)
- Gouverneurs
  - Performance/Powersave/Ondemand
  - NetSched

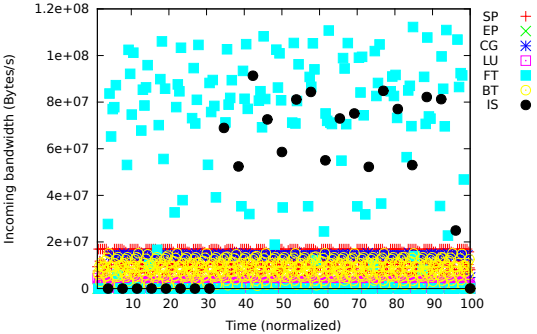
$$1.1B_1 \simeq 7.10^7 \text{ et } 0.9B_2 \simeq 3.10^7$$

Exemple d'exécution

Communication réseau en fonction du temps

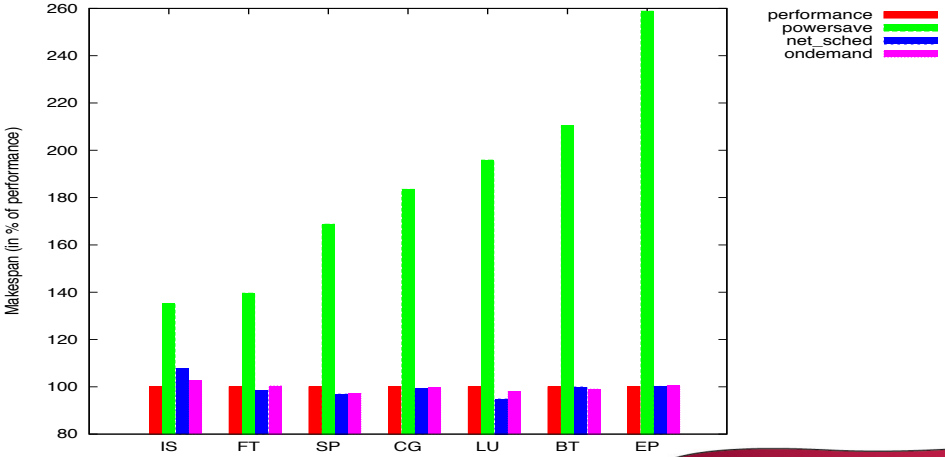


performance

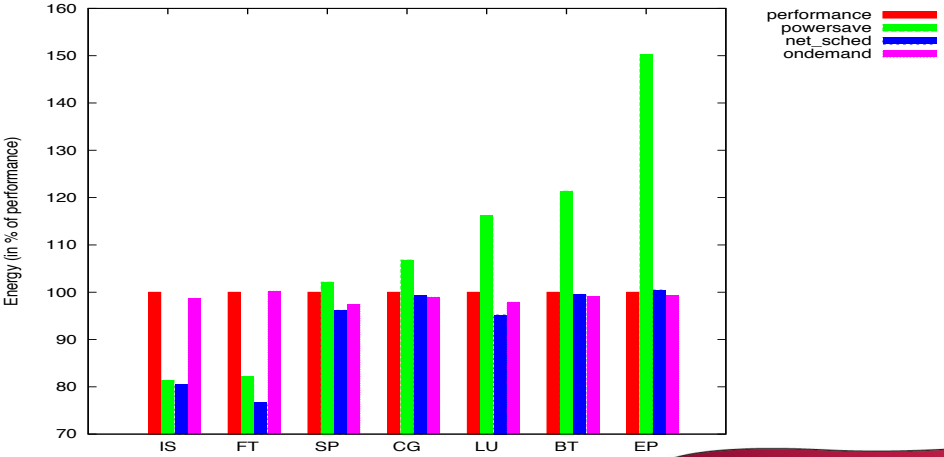


powersave

Résultats: Makespan



Résultats: Energy-to-solution



Plan

1 Boucle autonome
- Modèles et Métriques
- Mesures
- Outils d'évaluation

2 Décision
- Placement
- Fédération de clouds
- Data center in the box

3 Évolution, optimisation des noeuds
- Gros grain
- Moyen Grain
- Grain fin

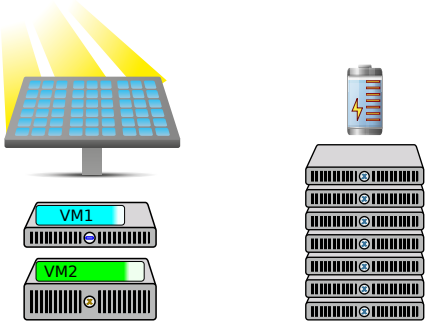
4 Et au delà

Le chaînon manquant entre les échelles

- Une activité “artisanale”
  - Un grand nombre d’intergiciels indépendants
  - Des manipulations humaines
- Vers une coopération décentralisée

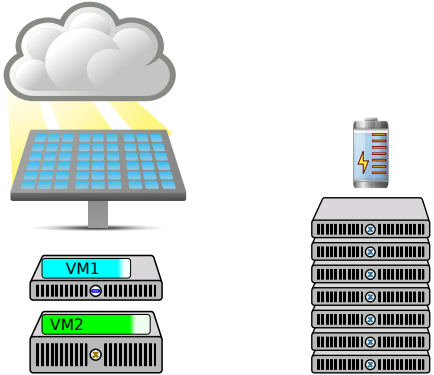
Coopération entre les échelles de décision

1. Situation initiale à l'équilibre



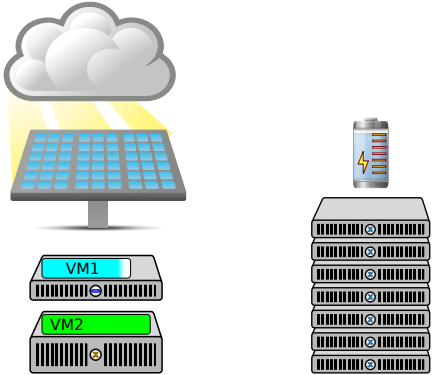
Coopération entre les échelles de décision

1. Situation initiale à l'équilibre
2. Baisse de la production solaire



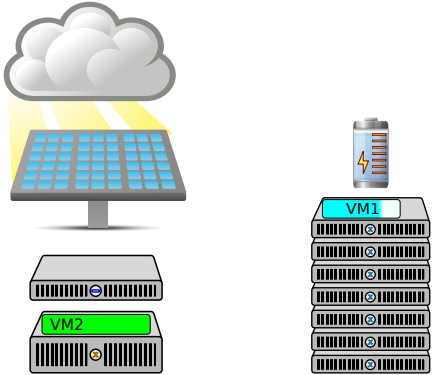
Coopération entre les échelles de décision

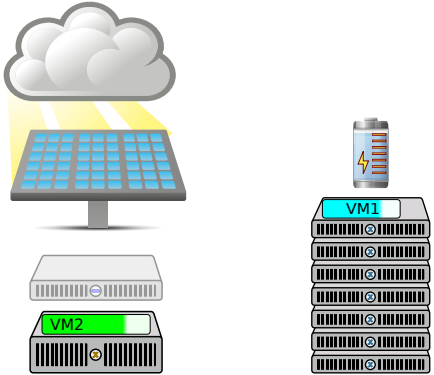
1. Situation initiale à l'équilibre
2. Baisse de la production solaire
3. Tâche non critique : DVFS agressif
3. Tâche critique : dynamisme indisponible



Coopération entre les échelles de décision

1. Situation initiale à l'équilibre
2. Baisse de la production solaire
3. Tâche non critique : DVFS agressif
3. Tâche critique : dynamisme indisponible
4. Tâche critique cherche position adéquate



Coopération entre les échelles de décision

1. Situation initiale à l'équilibre
2. Baisse de la production solaire
3. Tâche non critique : DVFS agressif
3. Tâche critique : dynamisme indisponible
4. Tâche critique cherche position adéquate
5. Extinction d'une machine
6. DVFS moins agressif