ENIB, module SHES-S8-EN

Consommation énergétique des technologies du numérique

(ressources sur `/home/TP/sujets/S8EN_Info/`)

Gag d’une série humoristique

- « The Cooper-Hofstadter Polarization »
  - « The Big Bang Theory »
    (season 1, episode 9, 17 March 2008)

Dans la « vraie » vie

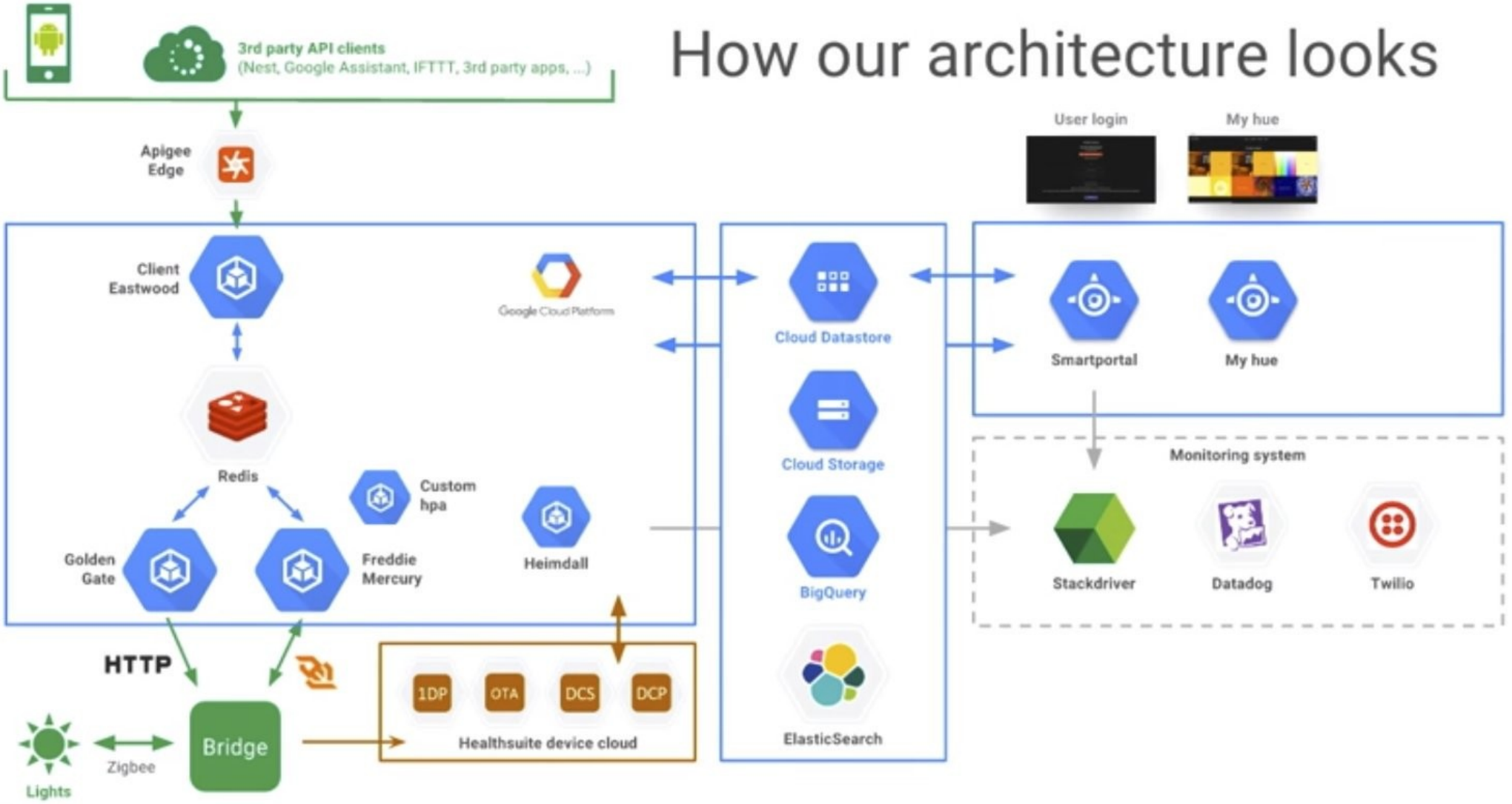
- Un produit de consommation courante
  - Technologie Philips Hue, produite en 2012

    (https://en.wikipedia.org/wiki/Philips_Hue)

- « Kubernetes and Google Container Engine » (Google Cloud Next '17)

  4:08 → 19:34, George Yanni,
  head of lighting for Philips Hue

Dans la « vraie » vie



Exemples informels

- Compteur de consommation dans le navigateur

  https://theshiftproject.org/en/carbonalyser-browser-extension/

  - Vidéo « Peru 8K HDR 60FPS (FUHD) », Apr 10, 2017
    - 5m37, 406 MB, 0.096 kWh, 50 gCO2e,
      6 smartphones chargés,
      0.227 km en voiture
    - 59,280,383 vues, 5.691 GWh,
      2964.019 tonnes CO2e,
      (300 Français·an de 2019, importations incluses 54 %)
      355.682 millions de smartphones chargés,
      13.457 millions de km en voiture

Exemples informels

- Courrier électronique en mars 2020

  Pièce jointe (vidéo) 4550036 octets

  Message complet avec l'encodage 6289459 octets

  1051 destinataires (tous@enib.fr + etudiants@enib.fr)

  ⇨ 6.16 GB stockés

- Courrier électronique en octobre 2020

  Pièces jointes (photos) 9191382 + 194403 + 4099187 octets, la plus « petite » fait 1280×1200

  Message complet avec l'encodage 18498537 octets

  1245 destinataires (tous@enib.fr + etudiants@enib.fr)

  ⇨ 21.45 GB stockés

Exemples informels

- Courrier électronique en décembre 2020

  Pièce jointe (document PDF) 2499719 octets

  Ce document contient des images à haute définition

  Message complet avec l'encodage 3847402 octets

  1245 destinataires (tous@enib.fr + etudiants@enib.fr)

  ⇨ 4.46 GB stockés

- Courrier électronique en janvier 2021

  Pièces jointes (documents PDF) 108703 + 4875331 octets

  Le premier document contient deux images à définition normale,
  le second document contient une image à haute définition

  Message complet avec l'encodage 6830380 octets

  1245 destinataires (tous@enib.fr + etudiants@enib.fr)

  ⇨ 7.92 GB stockés

Exemples informels

- Au total, 40 GB stockés par mégarde sur moins d'un an

  (juste quelques exemples, bien d'autres cas ont dû se présenter)
  (seulement à la toute petite échelle de notre établissement)

- Cas opposé : courrier électronique en novembre 2020

  Message du BDE (don du sang) 7653 octets seulement

  Lien vers informations complémentaires

  1245 destinataires (tous@enib.fr + etudiants@enib.fr)

  ⇨ 9 MB stockés seulement

  Ce message ne contient pourtant pas mille fois moins d'information utile que les précédents !

Exemples informels

- Invitation récurrente à exprimer un souhait dans un fichier de traitement de texte fourni
  - Réponse avec le fichier de traitement de texte 203428 octets
  - Réponse simple dans le message 1006 octets
  - Même usage à la fin (noter les quelques mots de la réponse pour la suite du processus)

⇨ Facteur deux cents dans les moyens utilisés (transport, stockage…) pour le même usage

Exemples informels

- Demande de dépannage en fournissant une image du message d'erreur plutôt qu'une copie du texte
  - Message avec l'image 42741 octets
  - Message équivalent avec juste le texte descriptif de l'erreur 1891 octets

⇨ **Facteur vingt-deux pour un usage moins utile**

On ne peut pas copier le message,
il faut le re-saisir pour effectuer une recherche par exemple

Infrastructures

- Surdimensionnement
  - Redondance, disponibilité, pics d'activité exceptionnelle
  - Indisponibilité, attente, saccades inacceptables !
    (secteur très concurrentiel)
- Trafic réseau mondial ⇗26 % par an
  - Mobile, ⇗60 % par an
  - Dans les *data-centres*, ⇗35 % par an
  - 8 à 10 milliards d'*e-mails* par heure (hors *spam*)
- Vidéo : 80% du trafic mondial (Netflix 15 %)
  - En 2019, Netflix ≈ 23 % trafic français (14 % en 2018)
    (Google ≈ 17 %, FaceBook ≈ 5 %)
  - *Streaming* ≠ télévision !
- Stockage *data-centres* mondiaux ⇗40 % par an

Consommation énergétique

- Part de l'énergie primaire mondiale
  - 4 % en 2013, 5 % en 2019
  - Projections de 7 % à 9 % en 2025
  - Croissance exponentielle, ⇗6.2 % par an (2015-2019)

- Gaz à effet de serre liés au numérique mondial
  - 2.9 % en 2013, 3.5 % en 2019 (aviation civile : 2.5 % en 2018)
  - Projection à 2025 : de 5.5 % à 6.9 % (automobiles : 8 % en 2018)

[Les sources sont indiquées sur
https://www.enib.fr/~harrouet/Data/Courses/S9ERI_Info.pdf]

!?!?! DÉMATÉRIALISATION !?!?!

- Quasi-impalpable pour l’usager
  - mais repose bien sur du matériel
    - de stockage, de calcul, de transmission, d’interaction...
  - qu’il faut
    - **produire/renouveler**, acheminer/installer, entretenir, alimenter, refroidir, retraiter…
- Impacts difficiles à quantifier
  - données réelles connues par les principaux acteurs
    - très délicat en termes d’image de marque
    - les études peuvent difficilement être indépendantes (résultats souvent controversés)

Bloatware (« Grogiciel »)

- Effet « rebond » de l’amélioration du matériel
  - Moyens de bas niveau toujours plus efficaces
    mais usage final toujours plus lent/lourd !
  - Ex : navigateurs ⇗ mais applications web ⇘
- Dédale de surcouches, de dépendances...
  - Conséquences sur les performances
  - Conséquences sur la maintenabilité
- Matériel/énergie moins chers que la réflexion !

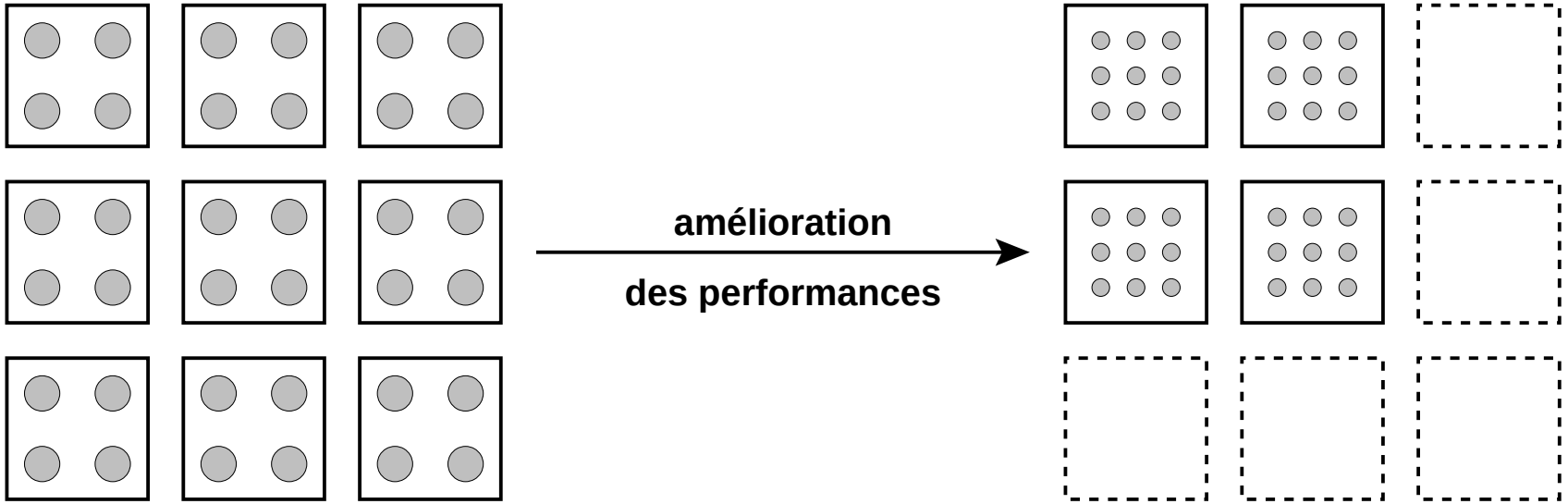
Performance ? Énergie ?

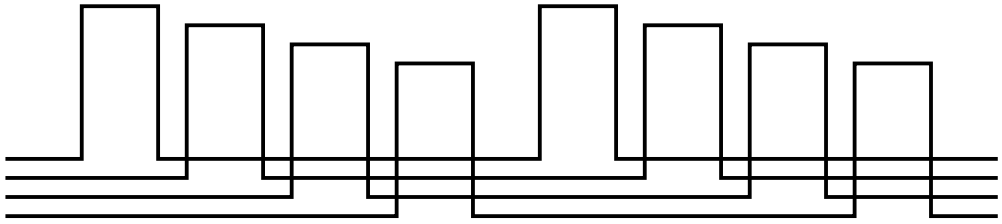
- Chandler Carruth : Google, clang/llvm
  « Efficiency with Algorithms, Performance with Data Structures », CppCon 2014
  - 4 min 12 s : mobile, 6 min 40 s : data-centres
  - 5 min 20 s : race to sleep!
  - 9 min 50 s : control over performance
  - 21 min 29 s → 27 min 53 s
    Surcoût éparpillé, indétectable par profiling !

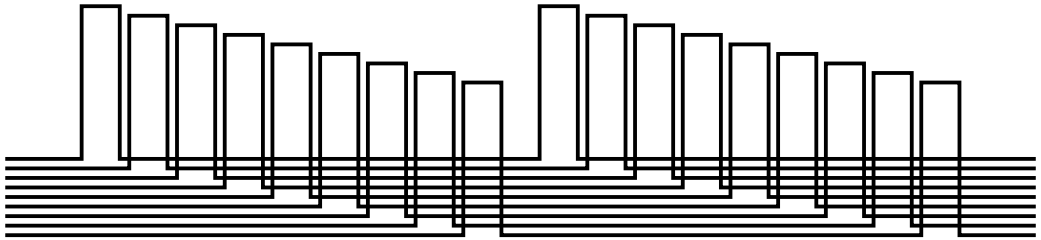
    ⇨ Porter une attention systématique à chaque opération !

Améliorer les performances ?

- Exemple : ressources des data-centres
  - Allocation des machines virtuelles sur les nœuds physiques selon la charge de travail







Modèle Intel-RAPL

- Estimation de la consommation d'un processeur (pas de tout le système)
  - Base de la plupart des outils d'analyse de la consommation
  - Lancer `./step_00.sh` et interpréter
- Diverses activités
  - Lancer `pyCpuEnergy sleep 10`
    ⇨ Constater la puissance au repos
  - Lancer `pyCpuEnergy sleep 60`
    Utiliser le poste (navigateur, PDF, éditeur, copies...)
    ⇨ Constater l'élévation de puissance
  - Lancer `pyCpuEnergy ./build.sh`
    ⇨ Constater l'élévation de puissance

Diverses charges de calcul

- Lancer `./step_01.sh`
  - Relever puissance au repos (baseline) : `PACKAGE+DRAM`
  - Relever pour chaque calcul
    - Coût en énergie : PACKAGE+DRAM – baseline×duration
    - Nombres d’opérations
    - Les rapports d’opérations et d’énergie pour en déduire les bénéfices

- Par rapport à l’utilisation usuelle ?

Parallélisation multicœur

- Lancer `pyCpuView &` et laisser ouvert
  - Explication de l'expérience avec l'appui de `pyCpuList`
- Lancer `./step_02.sh` et relever
  - Dans les deux cas : compute-bound et memory-bound
  - les coûts en énergie : PACKAGE+DRAM – baseline×duration
  - les gains en calcul selon le niveau de parallélisation
  - le bénéfice calcul/énergie selon le niveau de parallélisation

- Bilan ?

Performance ≠ raffinement

- Mauvaise interprétation de cette formule
  - « Premature optimisation is the root of all evil » Donald Knuth (1974)
    - Voir « Don’t Design for Performance Until It’s Too Late »
      - https://accu.org/index.php/journals/2136
- Ne pas « pessimiser » par défaut !
  - Pas besoin d’aller jusqu’à optimiser…
  - Il suffit d’en laisser l’opportunité au compilateur

Defensive Copy

- Garantir l'encapsulation malgré les déficiences du langage
  - C++ / Rust : déplacements, références sur const ou non-const
  - Python / Java : multiples copies profondes (sauf cas particulier)

    https://www.codejava.net/coding/java-getter-and-setter-tutorial-from-basics-to-best-practices

    http://www.javapractices.com/topic/TopicAction.do?Id=15

- Lancer et étudier le code de
  - `java DefensiveCopy bad` / `java DefensiveCopy good`
  - `./prog_no_defensive_copy demo` / `./no_defensive_copy demo`

- Lancer `./step_03.sh` et relever
  - Comparer les coûts en temps et en énergie

Exigences du matériel moderne

- Mémoires caches, prefetch, instructions vectorielles, prédictions de branchements…

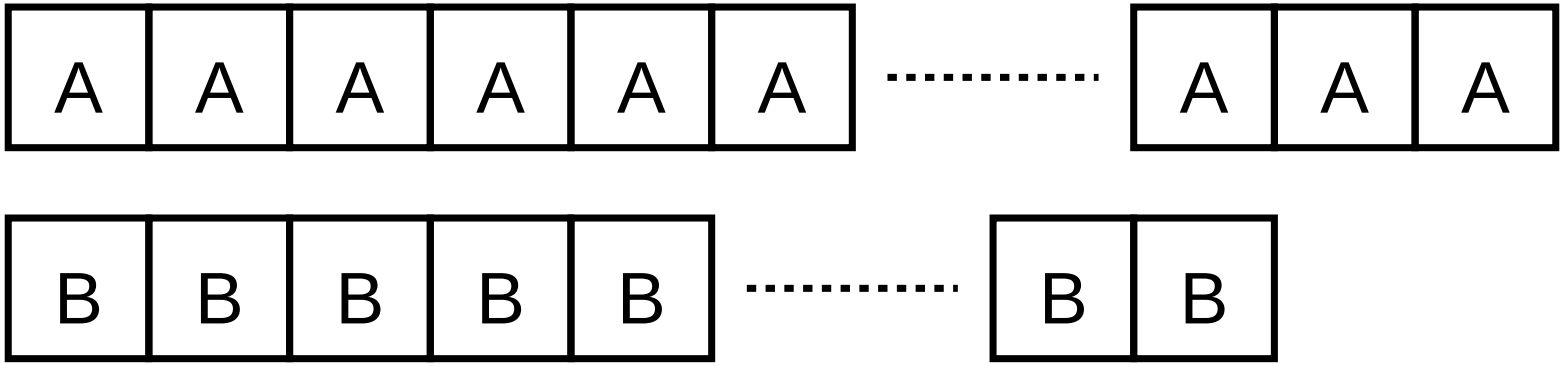
  ⇨ Privilégier les accès et les calculs réguliers

- Préoccupations de Modern C++ et Rust
  - Déterminer le plus possible à la compilation
    - Manipuler des types concrets, fonctions inline, lambda-closures, template, style fonctionnel...
    - Permet les optimisations, c’est à dire la reformulation selon des motifs réguliers

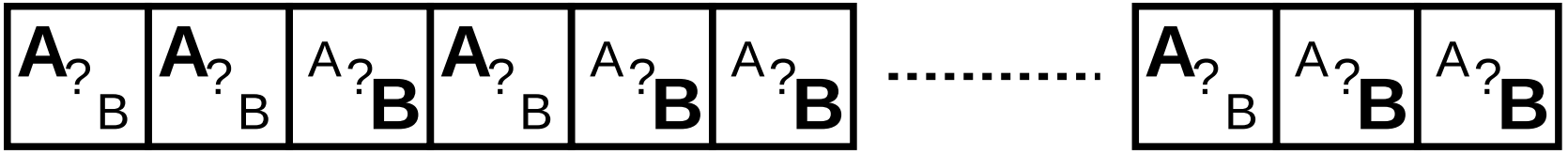
  ⇨ Éviter les indirections lors de l’exécution !

Paradigme de programmation

- Exemple : traitements simples sur deux types
  - Solution simple : polymorphisme statique
    Mémoriser et traiter les données par type

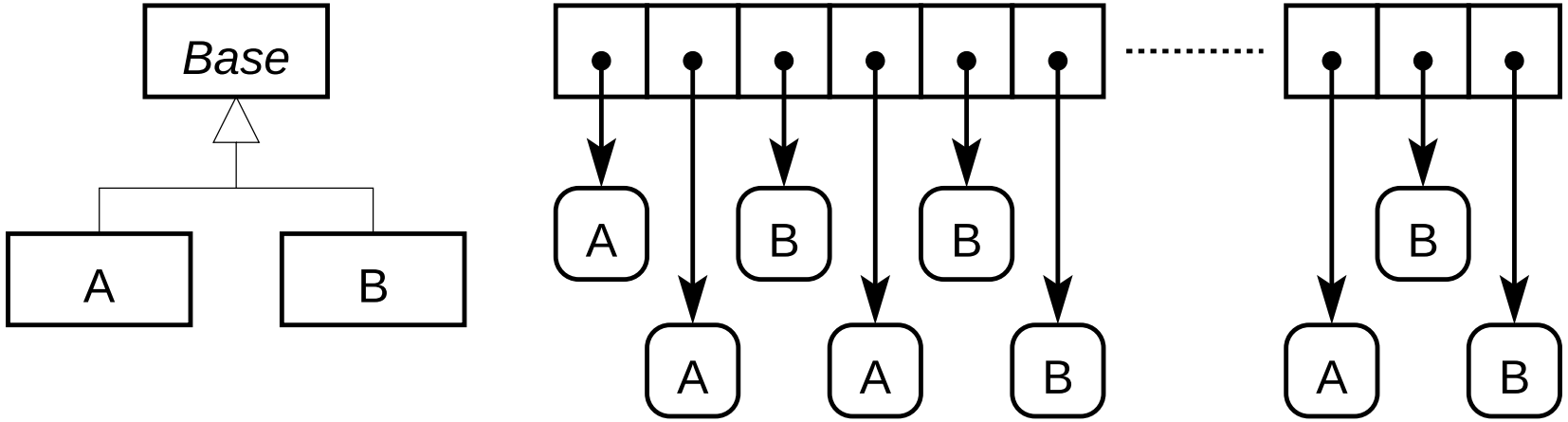


  - Si classement impossible, test d'une propriété



Para*dogme* de programmation

- Exemple : traitements simples sur deux types
  - Autre solution : polymorphisme dynamique (POO)



Para*dogme* de programmation

- Lancer `./step_04.sh` et relever
  - Comparer les coûts en temps et en énergie

⇨ « Inheritance Is The Base Class of Evil »

Sean Parent, Adobe (Photoshop), GoingNative 2013

⇨ « Data-Oriented Design (Or Why You Might Be Shooting Yourself in The Foot With OOP) »

http://gamesfromwithin.com/data-oriented-design

Communication réseau

- « Modeling The Power Consumption of Ethernet Switch », Hossain et al. (2015)

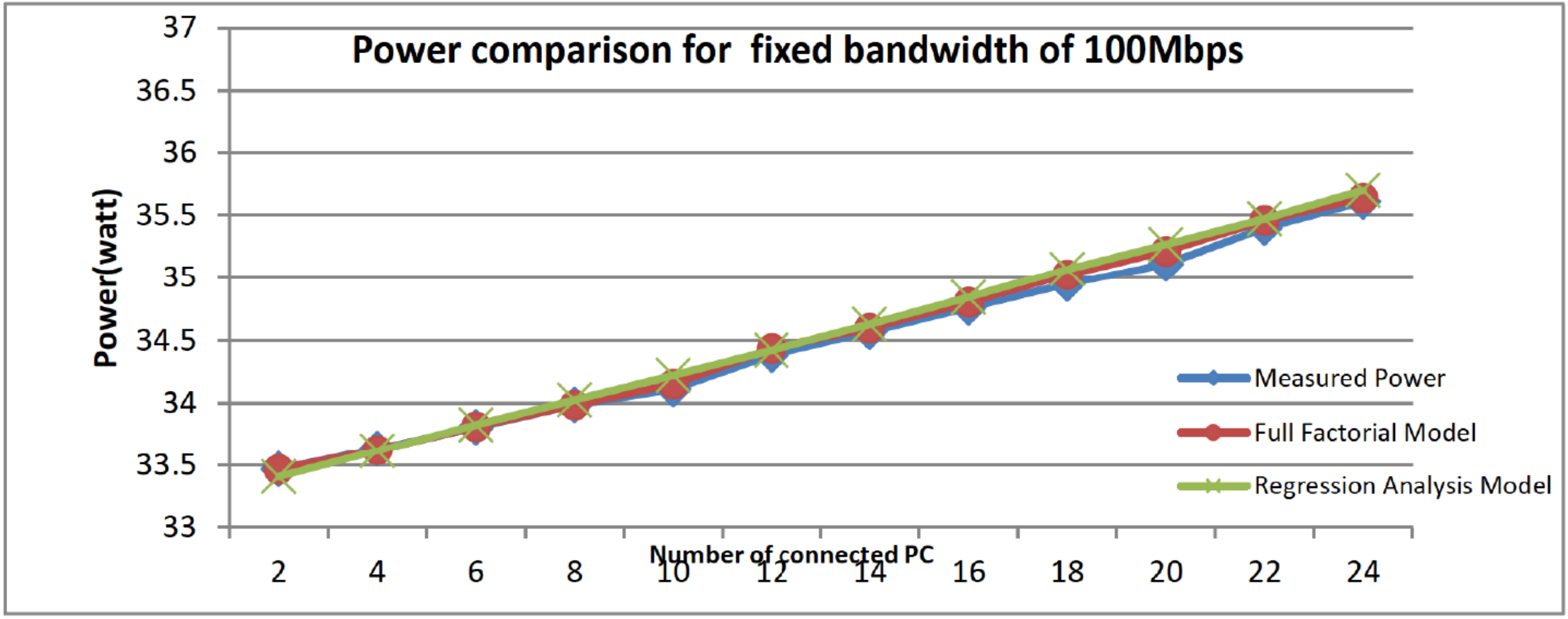


Figure 5: Power consumption comparison for fixed bandwidth of 100Mbps

Communication réseau

- « Modeling The Power Consumption of Ethernet Switch », Hossain et al. (2015)

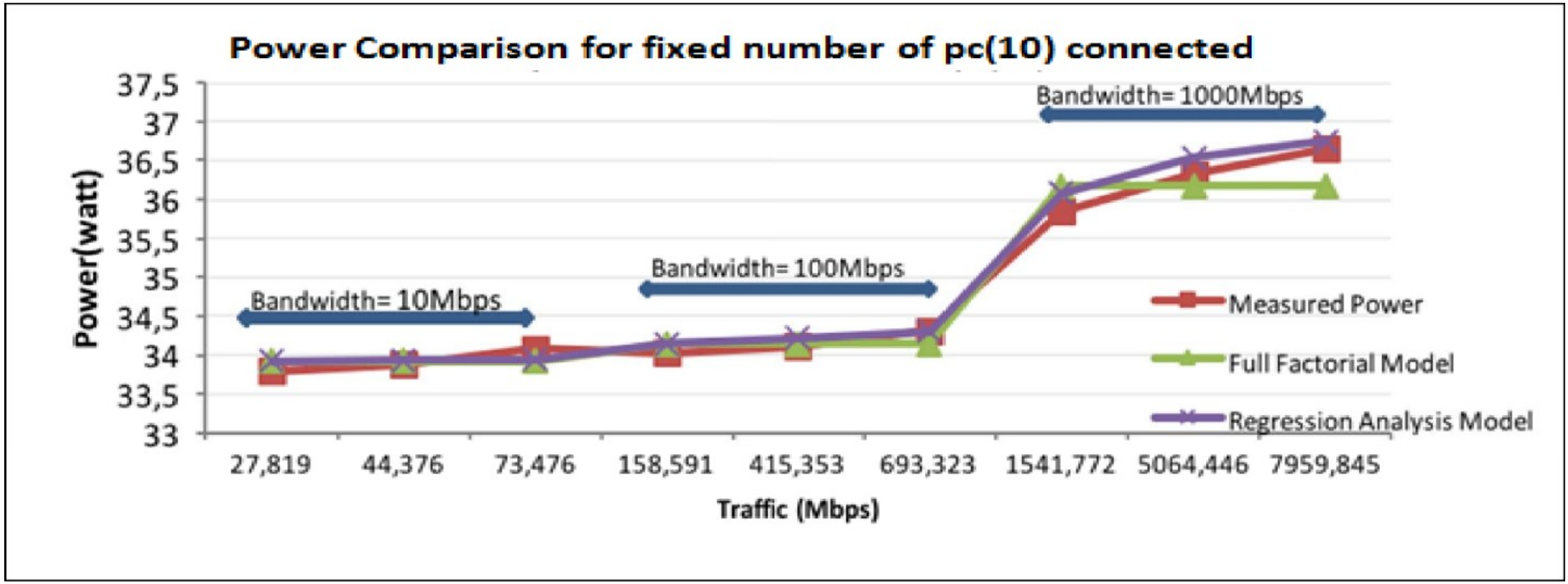


Figure 6: Power consumption comparison with change of traffic for fixed number of connected PC

Communication réseau

- Switch 24 ports en salle B013
  - 20.5 W au repos, postes éteints
  - 26.1 W au repos, postes allumés
  - 27.1 W transfert intensif entre tous les postes

- Petit switch domestique :
  - 1.3 W au repos, + 0.5 W par port qui télécharge

Communication réseau

- « Box » domestique :
  - Sans Wi-Fi : repos 15W, gros téléchargement 16W
  - Avec Wi-Fi : repos 18W, gros téléchargement 19W
  - ≈ 1 % consommation électrique (France, sans Box-TV)
    - Nombre de ménages en France en 2016 : 29.2 M

      https://www.insee.fr/fr/statistiques/4277630?sommaire=4318291
    - Consommation électrique en France en 2019 : 473 TWh

      https://www.edf.fr/groupe-edf/espaces-dedies/l-energie-de-a-a-z/tout-sur-l-energie/l-electricite-au-quotidien/la-consommation-d-electricite-en-chiffres
    - Box avec Wi-Fi en continu / an : 18 W × 24 h × 365 = 157.68 kWh
      Sur l'ensemble des ménages : 29.2 M × 157.68 kWh = 4.6 TWh
      (0.97 % de la consommation totale)

Communication réseau

- 1000 blocs de 16*1024 entiers de 32 bits
  - Format textuel : 6 caractères par entier
  - Format binaire : 4 octets par entier
  - Durées théoriques à 100 Mb/s ? à 1000 Mb/s ?
- Sur poste serveur :
  - `pyCpuEnergy -u 9876`
  - `./prog_tcp srv 9878 9877 9876`
- Sur poste client :
  - `pyCpuEnergy ./prog_tcp txt` *`server_host`* `9878 1000`
  - `pyCpuEnergy ./prog_tcp bin` *`server_host`* `9877 1000`
- Comparer la durée et l'énergie

Conclusion

- Corrélation entre efficacité en temps et en énergie
  - Pas nécessairement linéaire
- Les ordres de grandeur sont significatifs
- Pas de solutions systématiques
  - Évaluer l’adéquation d’une solution à un problème
  - Connaissances de base sur les matériels, les systèmes