ENIB, module S9-ERI

Les développeurs peuvent-ils encore limiter les ravages des technologies du numérique ?

Force est de constater…

- Enquêtes et rapports cités dans les médias
  - « *The Cloud Begins With Coal* »
    - Mark P.Mills, Digital Power Group, August 2013
  - « *Lean ICT* – Pour une sobriété numérique »
    - The Shift Project, Octobre 2018
      https://theshiftproject.org/article/pour-une-sobriete-numerique-rapport-shift/
  - « Impact environnemental du numérique :
    tendances à 5 ans et gouvernance de la 5G »
    - The Shift Project, Mars 2021
      https://theshiftproject.org/article/impact-environnemental-du-numerique-5g-nouvelle-etude-du-shift/
  - « L'âge des low tech : vers une civilisation techniquement soutenable »
    - Philippe Bihouix, Seuil, Avril 2014
  - « La guerre des métaux rares :
    la face cachée de la transition énergétique et numérique »
    - Guillaume Pitron, Les liens qui libèrent, Janvier 2018
  - Travaux et communications de Françoise Berthoud
    - Groupement de Service EcoInfo, CNRS

Force est de constater...

- Croissance exponentielle du déploiement des technologies du numérique (et donc des moyens associés)

- Impact en terme d'environnement et au-delà (économique, géopolitique...)
  - Consommation d'énergie, de matière première, dévastation/pollution des sols, surexploitation/pollution de l'eau...
  - Épidémies, conflits, dépendance aux ressources stratégiques...

- Opposition fondamentale entre les prétendues vertus de la transition numérique et la réalité des choses
  - La démarche va complètement à l'encontre de la cause qu'elle prétend défendre

[sans parler de la santé, des troubles de l’apprentissage, des addictions, de la vie privée...]

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)

- Anecdotes
  - Vidéo haute définition en *streaming* ≈ 1000W
  - *Boxes Internet* ≈ 1 % consommation électrique (France)
    (sans compter la fonctionnalité *Box-TV*)

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*)

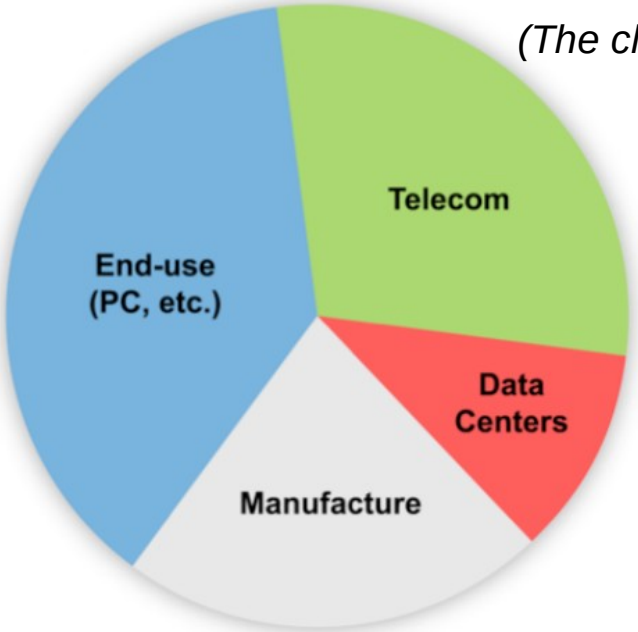
- Vidéo : 80% du trafic mondial (Netflix 15 %)
  - En 2019, Netflix ≈ 23 % trafic français (14 % en 2018)
    (Google ≈ 17 %, FaceBook ≈ 5 %)
  - *Streaming* ≠ télévision !

- Stockage *data-centres* mondiaux ⇗40 % par an

Production des équipements

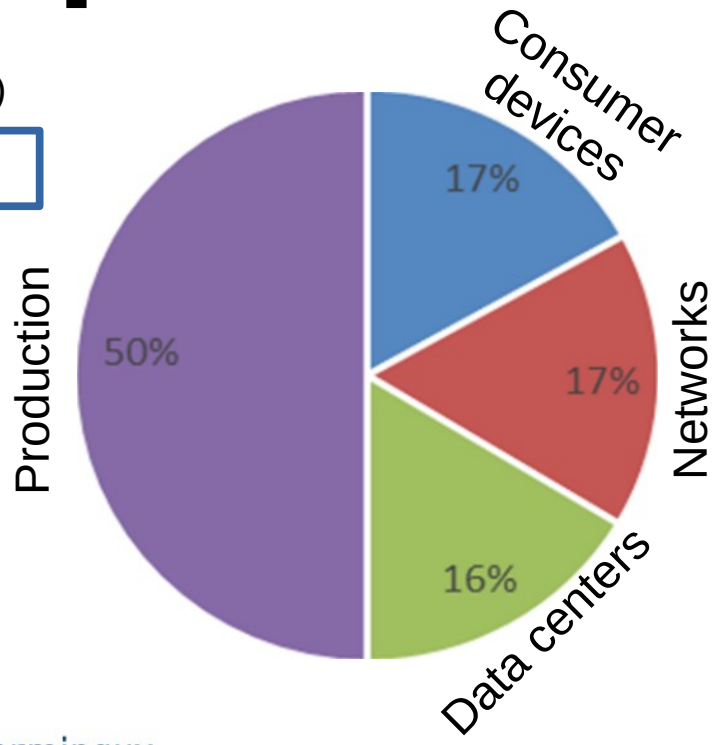
(The cloud begins with coal)

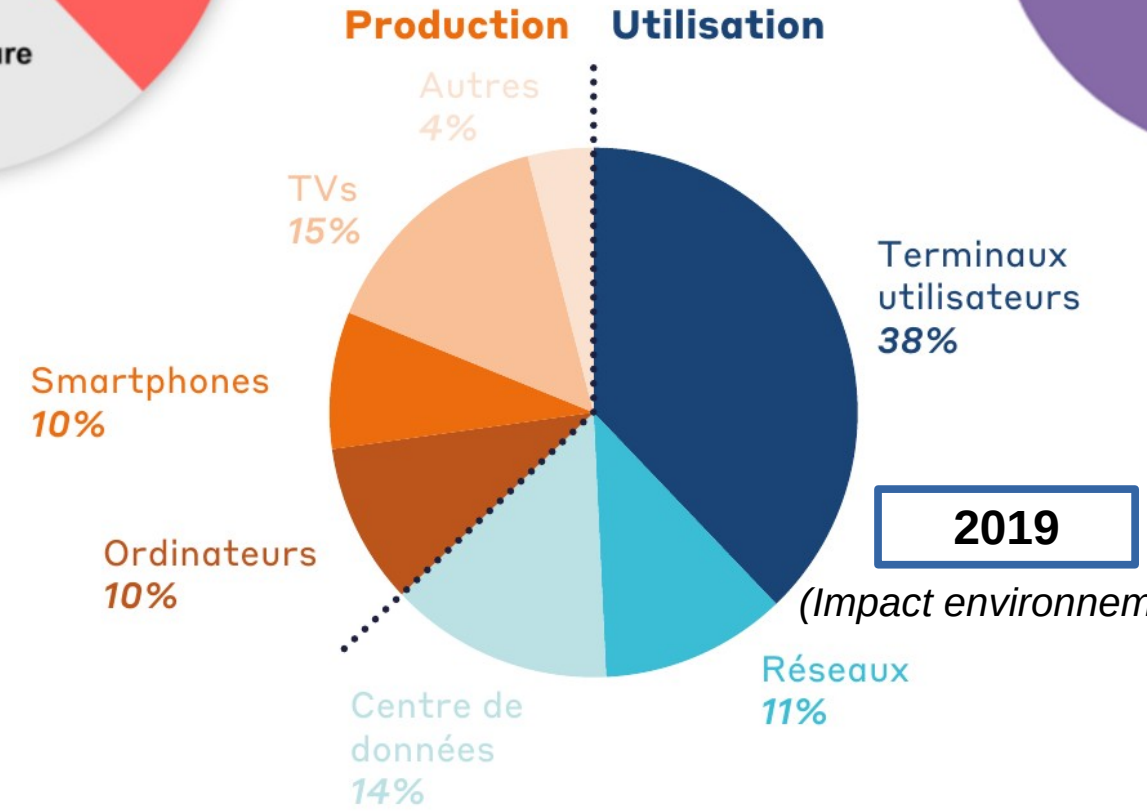
2013



(Lean ICT)

2017





2019

(Impact environnemental du numérique)

Production des équipements

- *Smartphones*
  - 90 % des GES du cycle de vie
    - 98 % en France (bouquet énergétique particulier)
  - Terminaux les plus produits
    - Renouvellement tous les 18 mois
    - 1.4 milliard en 2018 (160000 par heure !)
    - ⇗9 % du parc par an
  - Équipement principal dans les pays en développement

- *IoT* : tous les objets connectés
  - Explosion à venir dans les pays développés

Recyclage des équipements

- 20 % est collecté dans le monde (50 % en France)
  - Le reste ? (disséminé, pollution…)
- Collecte ≠ recyclage !
  - Reconditionnement (mais le matériel reste ancien)
  - Séparation, broyage, incinération, enfouissement…
- Au mieux, 3 % de métaux recyclés (sur collecte)
  - Usines spécialisées dans les *smartphones*
- Simple retard dans l’exploitation des ressources
  - Si la demande de consommation reste exponentielle

!?!?! DÉMATÉRIALISATION !?!?!

- Aucune inflexion des autres consommations
  - Papier, matériel, transport, tourisme…

    https://fr.wikipedia.org/wiki/Informatique_durable
    #Mauvaises_hypothèses_sur_le_rôle_des_TIC_pour_l'environnement
  - Discours vertueux très propice à l'effet « rebond »

- Accélération de notre consommation
  - De matériel surtout, imposée par le logiciel
  - Des services omniprésents
  - Impacts négatifs difficilement perceptibles à l'usage
  - Paradoxe : les équipements terminaux semblent de moins en moins consommateurs durant leur usage

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 !

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

Exigences du matériel moderne

- Mémoires caches, *prefetch*, instructions vectorielles, prédictions de branchements...

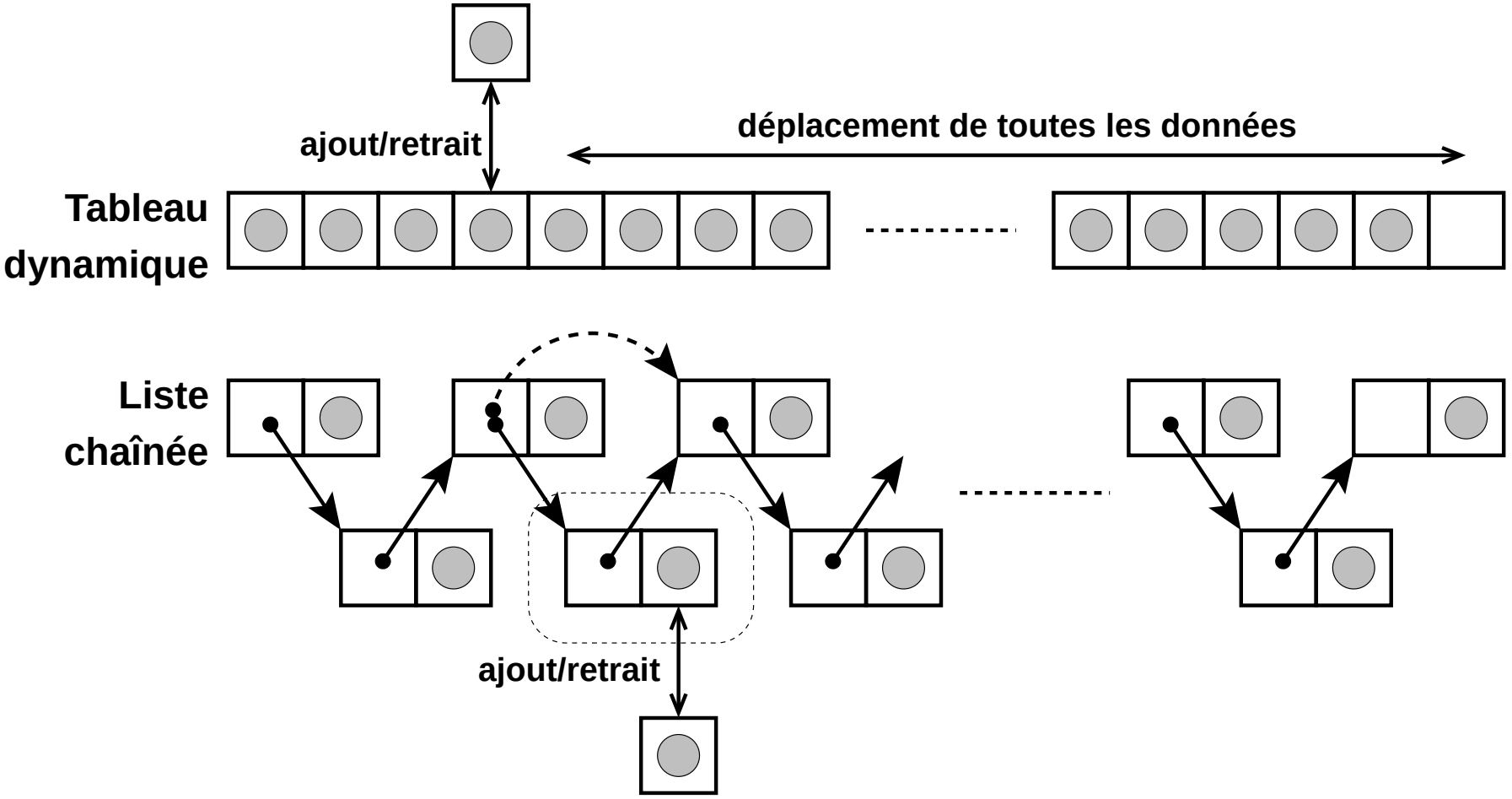
  ⇨ Privilégier les accès et les calculs réguliers

- 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 !

Structures de données

Exemple : « Software Development for Infrastructure »
Bjarne Stroustrup, IEEE, Computer, Jan. 2012



⇨ Tableau largement plus efficace que la liste !
(connaissance du fonctionnement des machines !)

Choix des algorithmes

- Exemple :

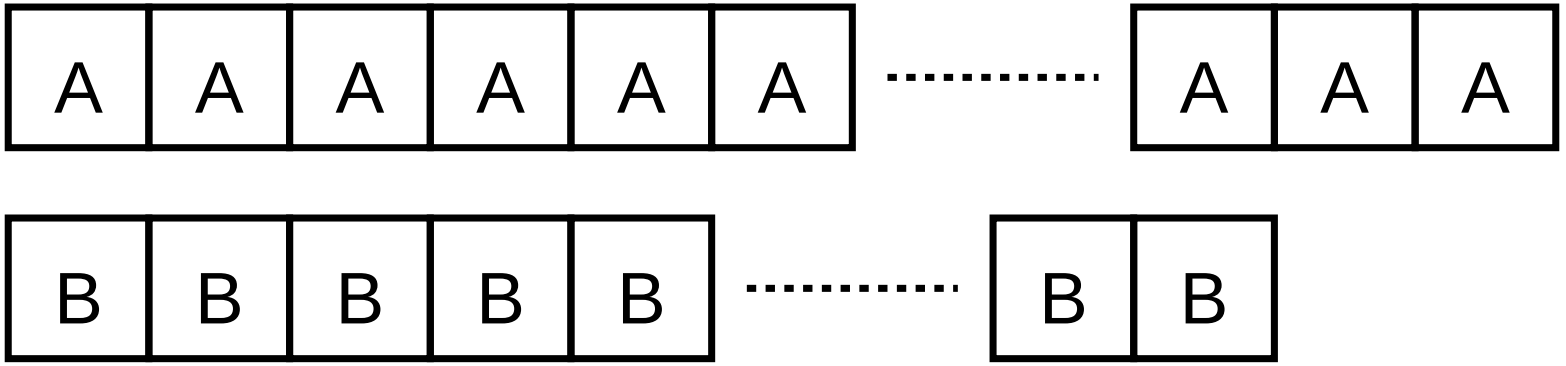
  « *‘Allegro’ Means Both Fast and Happy. Coincidence ?* »,

  Andrei Alexandrescu, NDC TechTown 2019,
  « *Speed Is Found In The Minds of People* »,
  Andrei Alexandrescu, CppCon 2019,

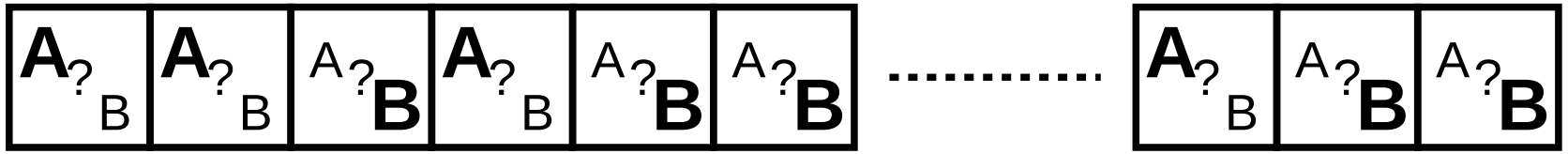
  - Variantes d’algorithmes de tri
  - Critères théoriques ⇗ mais performances ⇘
  - Des solutions théoriquement moins bonnes mais plus simples se révèlent meilleures en pratique !

Paradigme de programmation

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

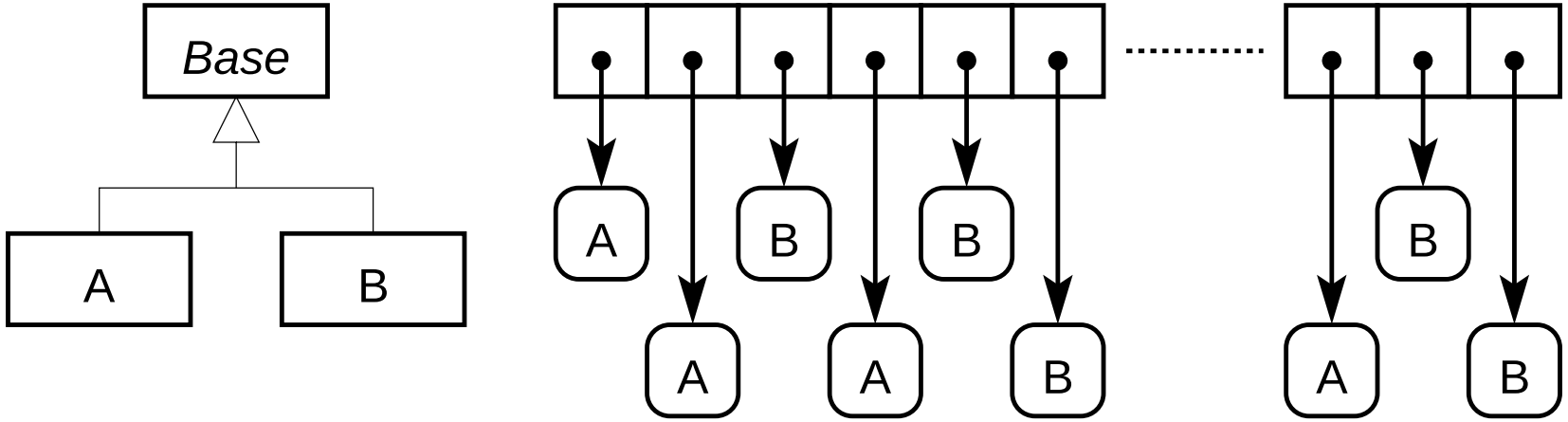


  - Si classement impossible, test d'une propriété
    ! coût × 4 en temps et en énergie !



Para*dogme* de programmation

- Exemple : traitements simples sur deux types
  - Autre solution : polymorphisme dynamique (POO)
    !!! coût × 30 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

Procédé de communication

- Échanges de données binaires
  - Production, transmission (taille fixe), exploitation
- Échanges de données textuelles (XML, JSON…)
  - Production, **formatage**, transmission (taille **variable**), **analyse**, exploitation
    !!! coût × 5…15 en temps, × 8…50 en énergie !!!

- Exemples :
  - HTTP → WebSocket : entêtes binaires et légers
  - CBOR/RFC7049 : JSON binaire (pas besoin de Base64)
  - Google Protobuf : messages binaires, structurés, compilés
  - HDF5 : fichiers binaires, extensibles, auto-descriptifs
  - …

Bases de données

- Exemple : 3 tables liées (qq 100k enregistrements)
  - Un « gros » poste : ancienne station de travail multi-cpu
  - Un « petit » poste : ordinateur portable moderne
  - Reliés par un *switch* à 1 Gb/s
  - Tantôt serveur, tantôt client

- Solution standard : jointures, clefs primaires/étrangères

  ⇨ Prendre le temps de se documenter/former sur les rudiments

- Solution « naïve » : boucles de requêtes imbriquées :-(

  ⇨ Syntaxe d'un requête élémentaire + algorithmique

  !!! coût × qq 100 en temps et en énergie !!!

  (que ce soit ancien/moderne client/serveur)

Tendance

- Préoccupation moyennement significative...
  - Usage de C++ pour l'embarqué et les *data-centres*
  - *Emscripten/WASM : C++/Rust dans le navigateur*
  - *Mais des projets ambitieux qui n'aboutissent pas...*
    - Java ressemblant encore plus à C++ (projet Valhalla)
      - *value-types*, *generics* → *templates* (et non plus *auto-boxing*)
    - includeOS : nœuds de *cloud* sans OS
      - Directement l'applicatif en C++
        - démarrage en quelques ms, empreinte mémoire minimale
      - Modèle événementiel et asynchrone, sans ordonnanceur
- L'engouement se porte plutôt sur :
  - *IA*, *VR*, *cloud-gaming*, vidéo-8K sur le mobile, véhicule autonome, crypro-monnaie...

Conclusion

- Faire des choix raisonnables en ressources ?
  - Indispensable de **comprendre** les mécanismes sous-jacents des matériels, des infrastructures et des API
- Économie de réflexion → surcoût en ressources
  - Coût lié au problème / à la solution ?
  - Pas de recette miracle mais quelques précautions
    - Données compactes, classées
    - Visibilité du code par le compilateur, types concrets
    - Le strict nécessaire
    - Mesurer, comparer les solutions (*godbolt*, *google-benchmark*, *perf-counters*, *intel-rapl*...)

Mais...

Merci pour votre attention