

Sujets de projet M1 Informatique

Millian POQUET

2025-10-14

Millian POQUET

- Enseignement : systèmes d'exploitation, synchro et parallélisme
- Recherche : gestion de grands systèmes distribués, énergie, sobriété, reproductibilité

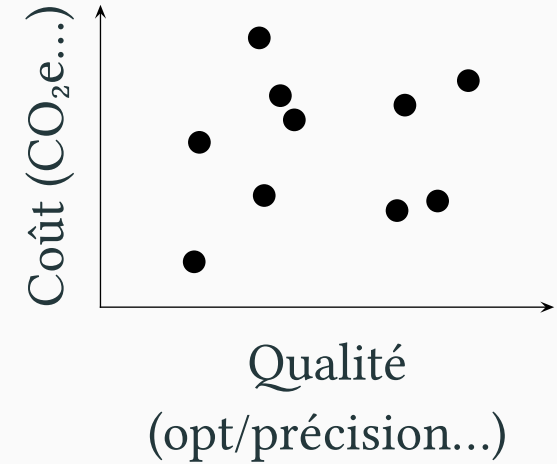
11 sujets de ~~sobriété~~ d'écoconception

- 4 liés enseignement
- 8 liés recherche

Projet libre de R&D

Idée : étudier un problème

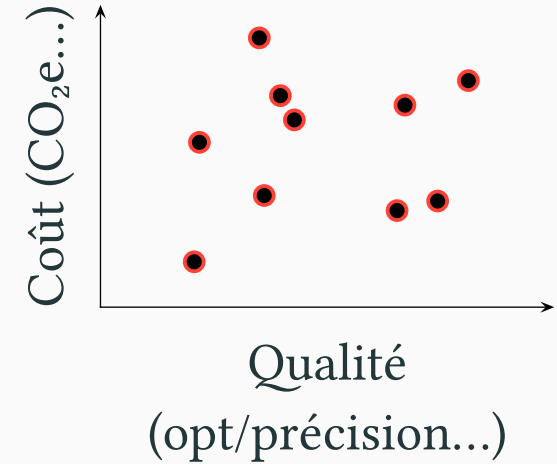
1. Déterminer technos / papiers existants
2. Développer des alternatives
3. Évaluer ces alternatives
4. Conclure



Projet libre de R&D

Idée : étudier un problème

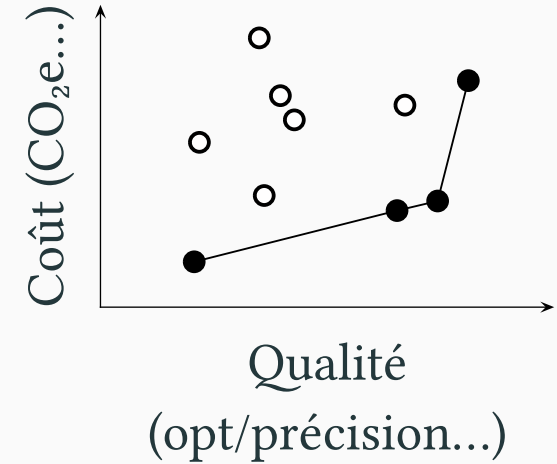
1. Déterminer technos / papiers existants
2. Développer des alternatives
3. Évaluer ces alternatives
4. Conclure



Projet libre de R&D

Idée : étudier un problème

1. Déterminer technos / papiers existants
2. Développer des alternatives
3. Évaluer ces alternatives
4. Conclure



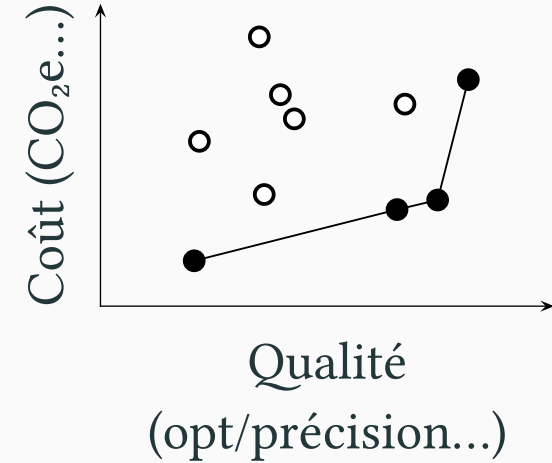
Projet libre de R&D

Idée : étudier un problème

1. Déterminer technos / papiers existants
2. Développer des alternatives
3. Évaluer ces alternatives
4. Conclure

Gestion du projet

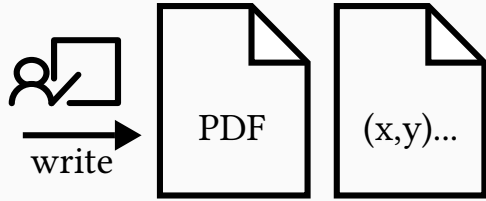
- Comme bon vous semble
- Suivi (qualité) via Git
- Pseudo cahier des charges au début
- Quelques réunions obligatoires



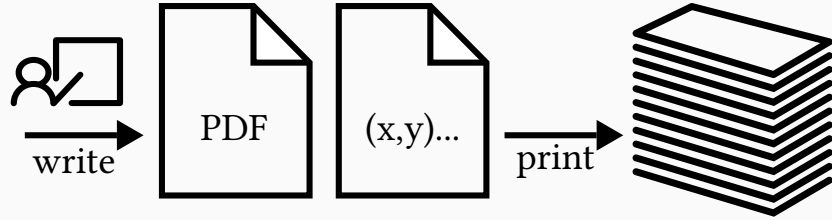
Contraintes

- Code → Apache-2.0
- Doc et données → CC BY 4.0
- Env logiciel → Nix
- Rédaction → Typst (LaTeX)
- Lang/Techno raisonnables

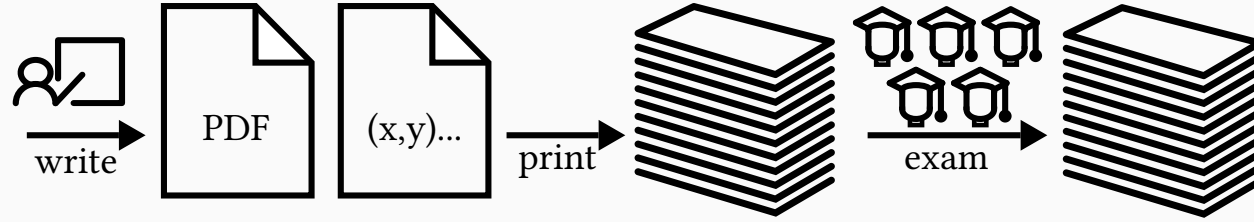
Hekzam (intro)



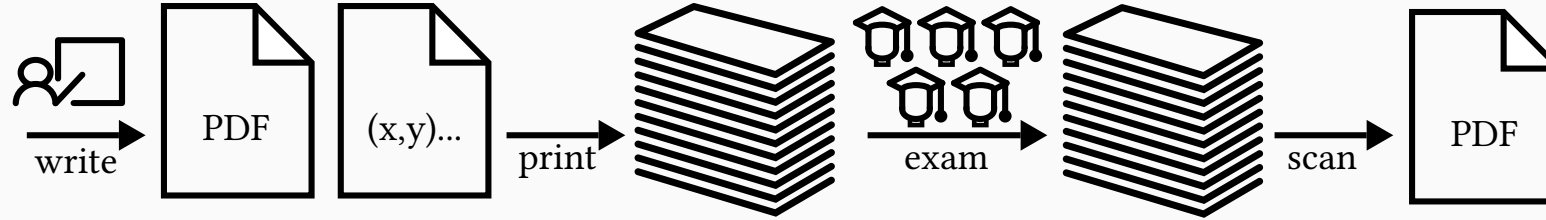
Hekzam (intro)



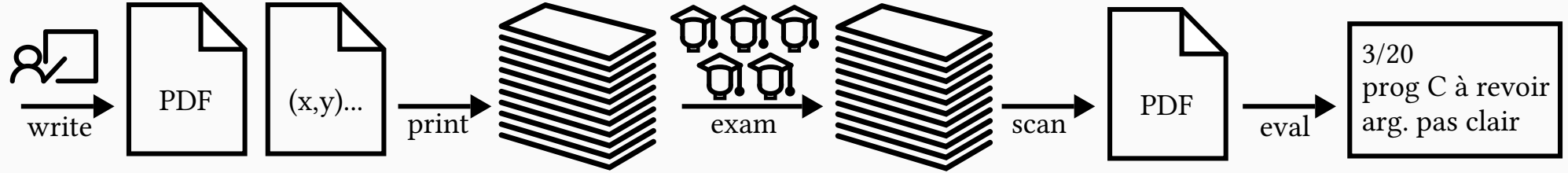
Hekzam (intro)



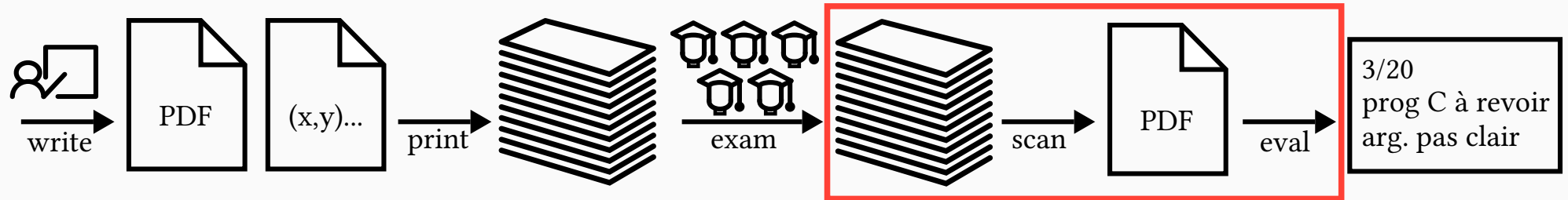
Hekzam (intro)



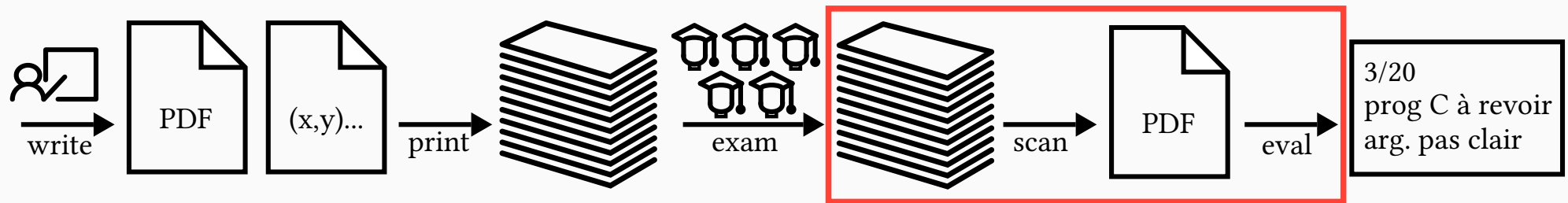
Hekzam (intro)



Hekzam (intro)



Hekzam (intro)



Traitement de copies complétées scannées

1. Comment *bien* extraire les saisies numériques ?
2. Comment *bien* transformer les images en chiffres ?
3. Comment *bien* attribuer les copies aux apprenant·e·s ?



N° étu

Nom

contenu de l'examen

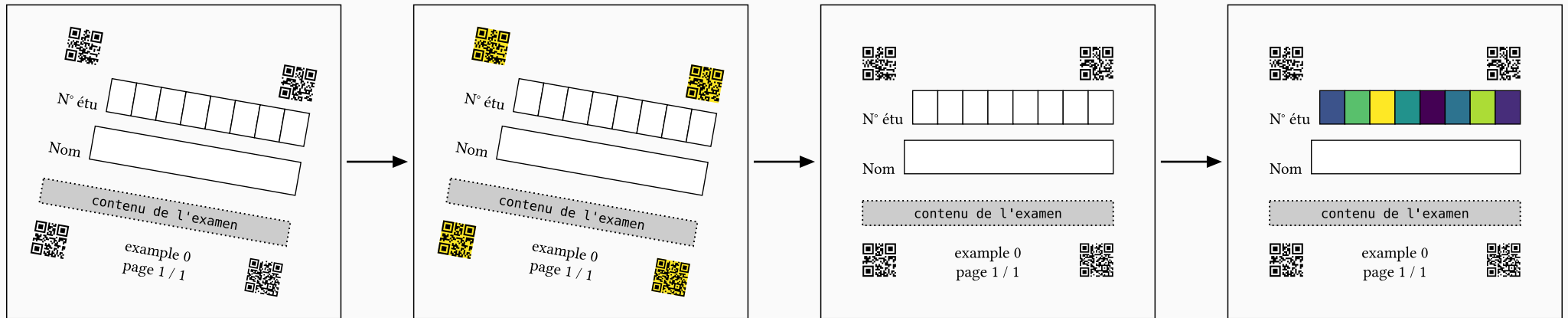


exemple 0
page 1 / 1



Sujet 1 — Hekzam : détection précise de cases dans un espace déformé

Scan *classique* : transformation affine (translation, rotation)

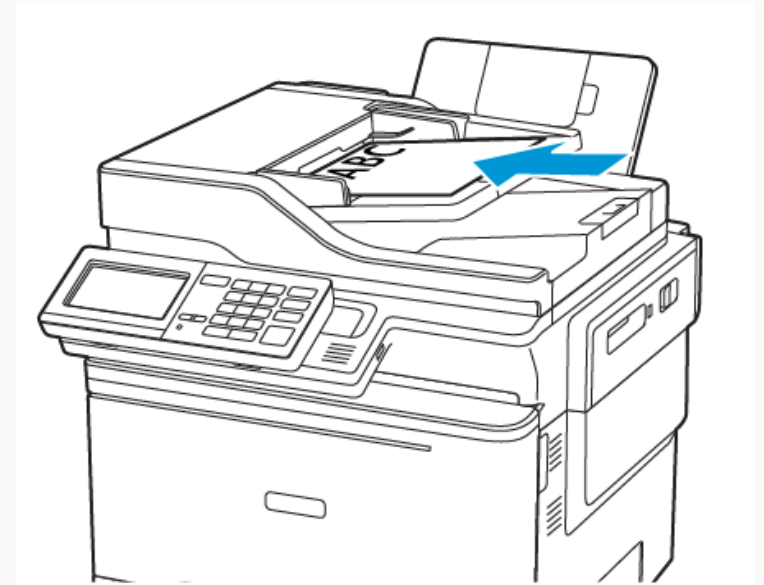


Sujet 1 — Hekzam : détection précise de cases dans un espace déformé

En pratique, on utilise un alimentateur automatique qui

- Mange rapidement les feuilles
- Bombe les feuilles lors du scan
- Ne conserve pas une vitesse de scan constante

La déformation est difficile à inverser !



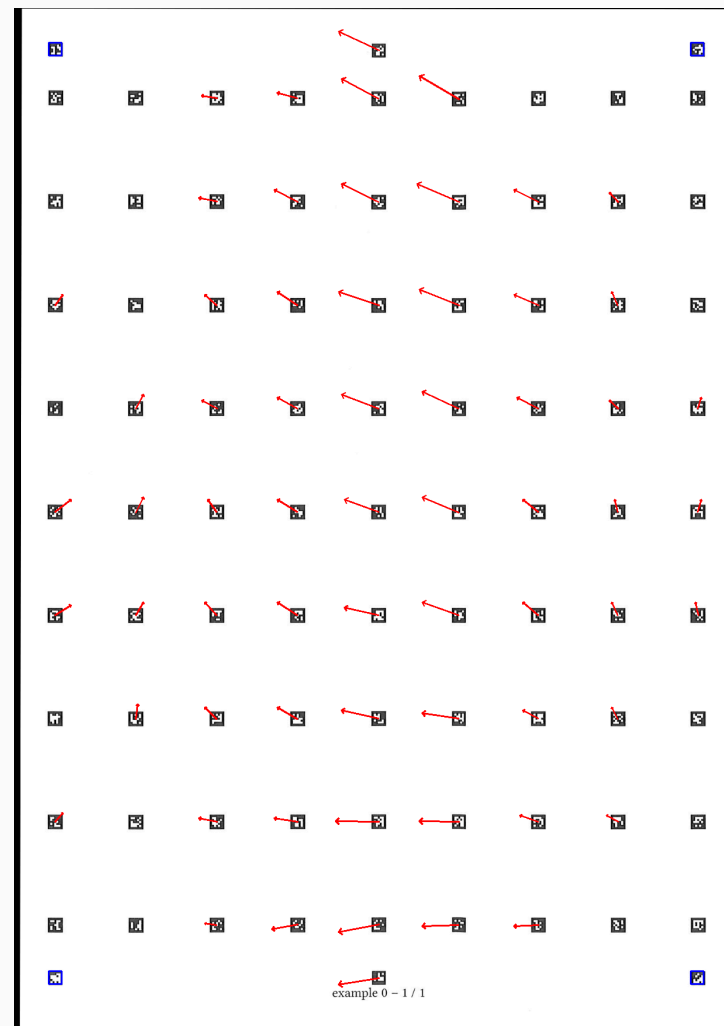
Sujet 1 — Hekzam : détection précise de cases dans un espace déformé

En pratique, on utilise un alimentateur automatique qui

- Mange rapidement les feuilles
- Bombe les feuilles lors du scan
- Ne conserve pas une vitesse de scan constante

La déformation est difficile à inverser !

À droite : différence entre observé et prévu après transformation affine de marqueurs (norme $\times 20$)



Sujet 1 — Hekzam : détection précise de cases dans un espace déformé

Problème : extraire le contenu des cases numériques avec une bonne précision

- Algo 0 : transfo affine seule (implem actuelle)
- Algo 1 : transfo affine suivie d'une détection de rectangles pour corriger déformations
- Autres : à déterminer par vous

Métriques

- Précision d'extraction des boîtes
- Temps total d'extraction

Contraintes technos

- C++
- OpenCV

Sujet 2 — Hekzam : ROC et réseaux de neurones frugaux

Problème : compromis temps/précision de prédiction de différents réseaux **frugaux**

- Petits CNN
- Petits perceptrons multicouche
- ...

Problème : quelle robustesse de ces réseaux à des erreurs d'extraction d'images ?

Métriques

- **Temps de prédiction** sur CPU peu puissant
- **Accuracy** de prédiction
- Espace mémoire nécessaire aux modèles
- Temps d'apprentissage sur CPU peu puissant

Dataset MNIST

 → 0

 → 5

Sujet 3 — Hekzam : fuzzy matching de numéros identifiants

Problème : matching entre liste de numéro étus et numéros *incertains* issus de scans

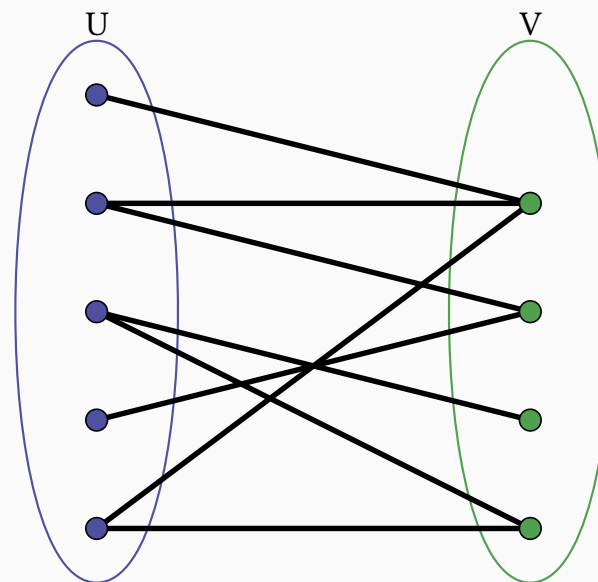
- Quel est le numéro étu le plus probable de chaque copie ?
- Pas de duplication ? ($n > 1$ copies pour 1 étu)
- Pas de copie manquante ? (0 copie pour 1 étu présent·e)

Questions scientifiques

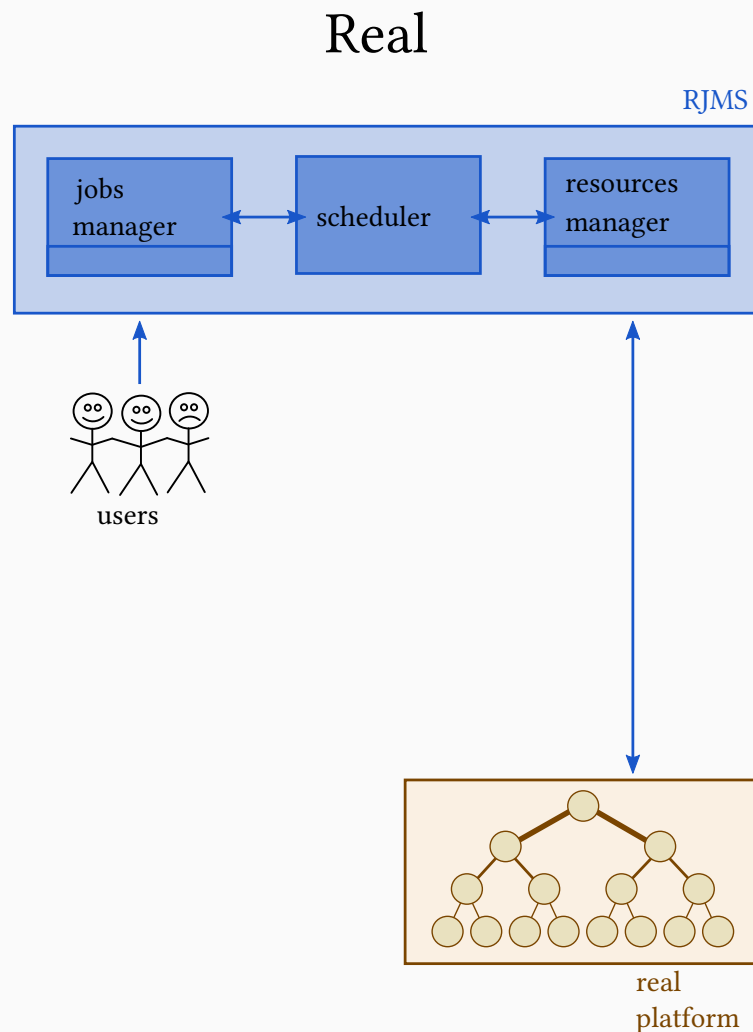
- Quels compromis qualité / perf des algos ?
- Quelle accuracy de ML est suffisante ?

Outils / techniques prévues

- Fuzzy matching
(Levenshtein/Hamming/... distance)
- Arbres spécialisés (BK trees...)

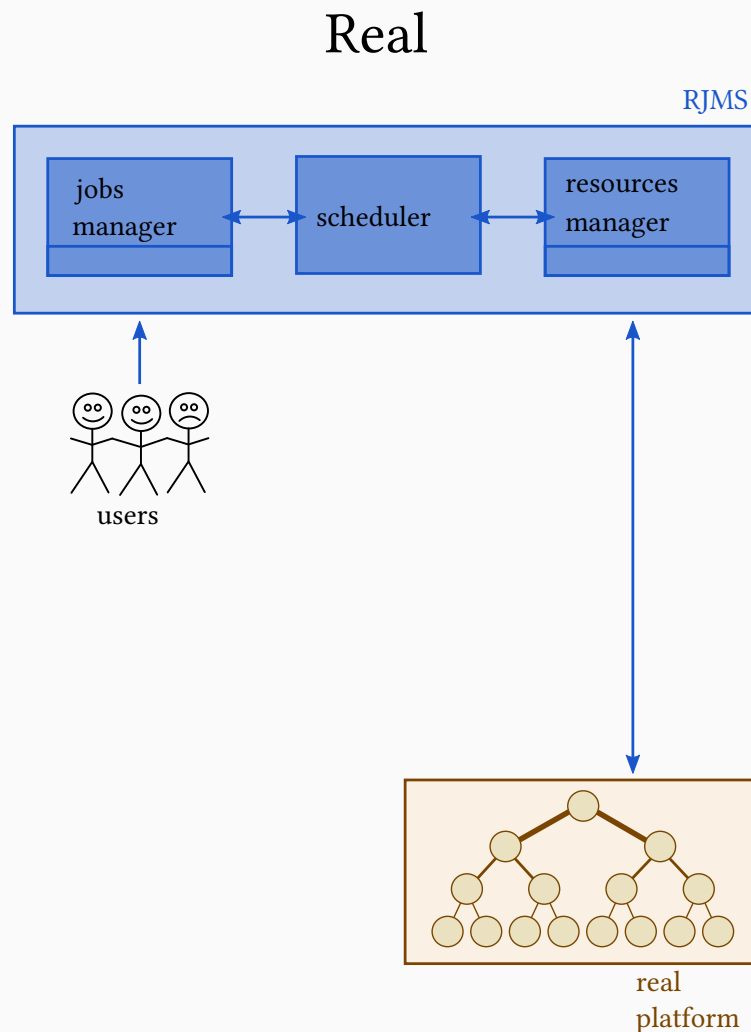


Gestion de ressources à grande échelle (intro)

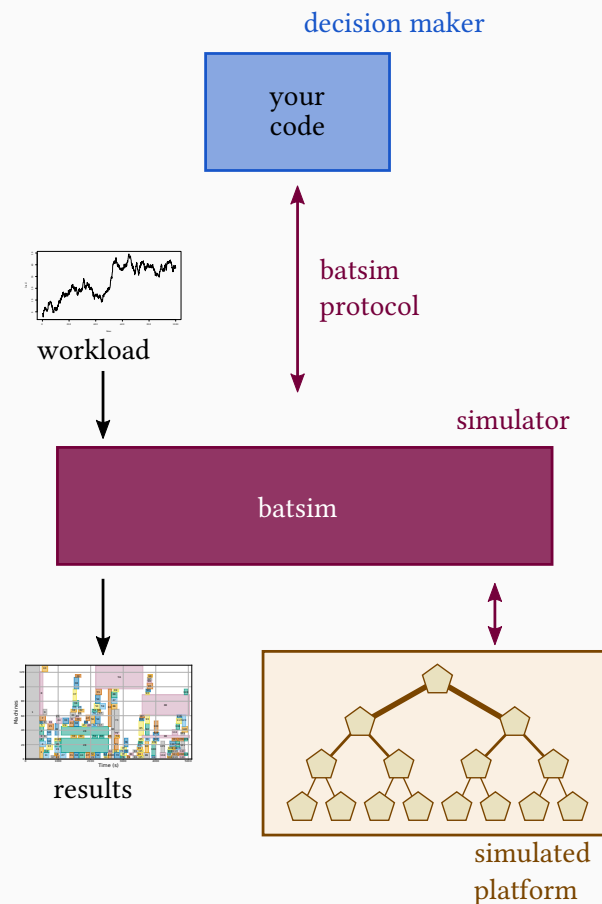


Batsim simulation

Gestion de ressources à grande échelle (intro)



Batsim simulation



Sujet 4 — Rusty Bataar

But : permettre d'étudier de **vraies impléms** d'algos de sched dans Batsim

- Existant à MAJ : Bataar entre OAR3 (Python) et Batsim
- À faire : pont entre OAR3 (Rust) et Batsim

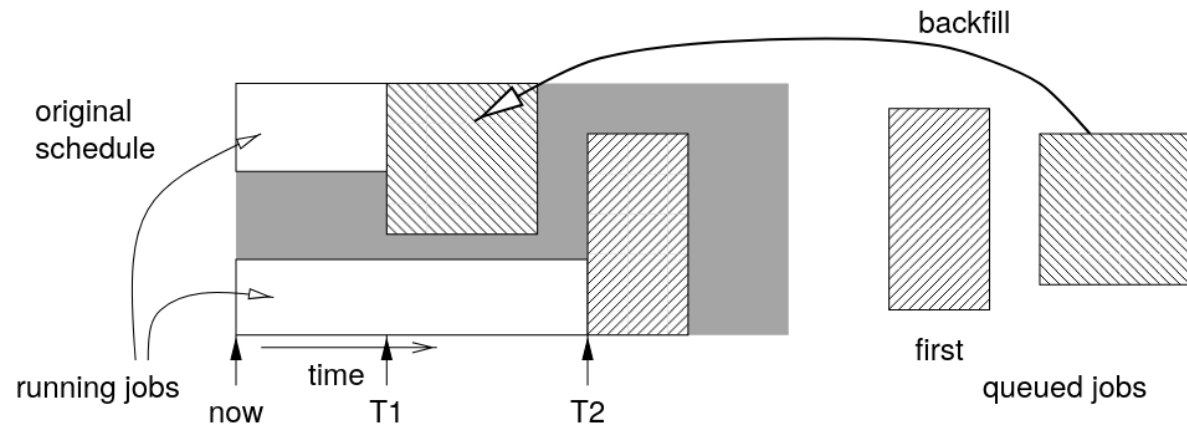
Sujet de ~~R&D~~ developpement

- Pas d'éval de perf
- Exigence sérieuse de tests

Sujet 5 — Batch Scheduling in Rust

Buts

- Développer un framework de scheduling Batsim en Rust
 - Structures de données (pour backfilling) réutilisables
- Éval perf des structures de données
- Comparaison perf VS existant en C++



Feitelson+Mualem, IEEE TPDS 2001

Sujet 6 — Structures de données pour ensemble de ressources en Rust

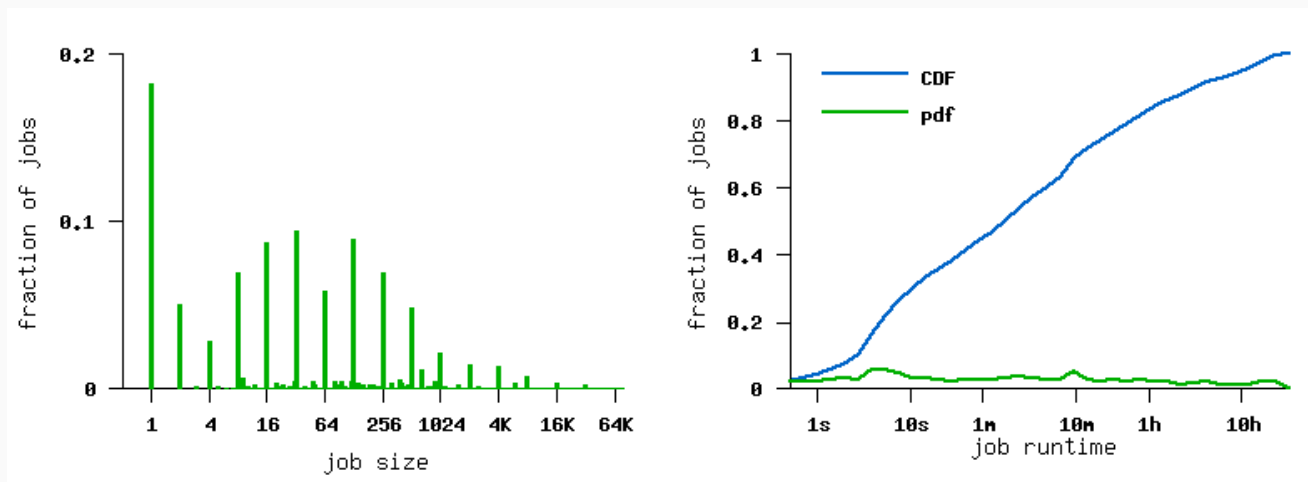
Buts

- Faire différentes implems du même type abstrait (ensemble d'entiers)
 - Arbres (BST, red-black tree, AVL...)
 - Table de hachage
 - Bitsets
 - Liste/vecteur d'intervalles
- Tester implems fonctionnelles et cohérentes
- Éval perf de toutes les implems
 - Selon le nombre de ressources max
 - Selon la fragmentation des ressources
 - Selon...

Sujet 7 — Outils d'analyse et de visualisation de traces d'exécution

Logiciels attendus

- Bibliothèque Python de visualisation de traces d'exécution
Grammar of Graphics (λ) attendue, à la `plotnine` ou `ggplot2`
- Reproduction de l'ensemble des visualisations de la Parallel Workloads Archive
<https://www.cs.huji.ac.il/labs/parallel/workload/>
- Nouvelles visualisations dédiées à l'analyse de comportement utilisateur
- Extension aux sorties de simulation Batsim



Nix

- Outil gestion de paquet (apt-like)
- Outil gestion d'environnements temporaires (virtualenv ou docker run-like)
- Outil construction de paquets (docker build-like)
- λ DSL pour définir paquets et environnements

Excellentes propriétés

- Traçabilité de **toutes** les données (sources, config et paramètres de chaque outil...)
- Reconstructibilité de **tous** les logiciels (lib Python, cpython lui-même, linux lui-même...)

Améliorations à étudier

- Optimiser taille des environnements produits
- Optimiser la construction des paquets

Sujet 8 — Optimiser la taille des environnements Nix

Cas d'étude : conteneurs Dockers utilisés pour faire une expérience scientifique

- Petits scripts de génération d'entrées en Python
- Simulateur en C++
- Scripts d'analyse et de visualisation en R

Problème : la taille des conteneurs est optimisé $\ll$ Nix $<$ naïf

Attendus

- Génération et optimisation de conteneurs via Nix via différentes méthodes
- Génération de conteneurs **très** optimisés en taille, *à la main* via différentes méthodes

Sujet 9 — Intégrer ccache dans Nix

ccache is a compiler cache

- générique à tout compilateur
- suppose que le compilateur est déterministe
- utilise un hash des fichiers entrées (et les autres entrées types flags et fichiers de config)

Nix a les mêmes hypothèses et hash les entrées, mais n'utilise pas ccache

→ Si on change 1 bit dans un énorme projet C++ qui ne nécessite de compiler qu'un seul objet, **tout** le projet doit être recompilé

Attendus

- Ressusciter l'intégration ccache dans Nixpkgs (code en Nix)
- Si besoin, modifier l'implémentation de Nix lui-même (code en C++)

Sujet 10 — Visualisation mémoire pour l'apprentissage et le debugging

Logiciels attendus

- Minidebugger en exécution pas à pas (ptrace + DWARF)
- Bibliothèque bas niveau d'extraction d'informations mémoire d'un processus (Linux)
- Codes de visualisation (langage haut ou bas niveau)

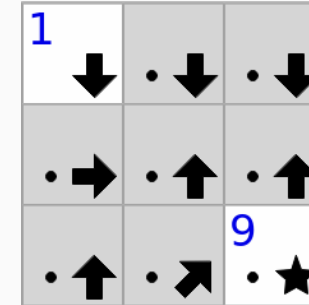
Exemples de visus attendues

- Où sont les variables en mémoire ?
- Évolution de la pile d'exécution à chaque bloc / appel de fonction
- Emplacement des *régions* mémoire d'un processus (piles, tas...)
- Droits d'accès aux pages mémoire
- Lien entre adresses virtuelles et physiques ; mémoire partagée

Sujet 11 — Signpost : analyse théorique et génération contrainte de grilles

Signpost : jeu type Puzzle

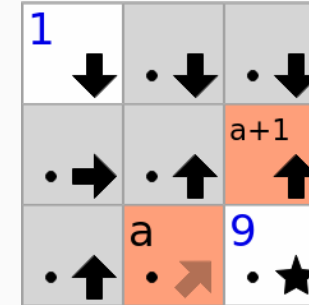
- Faire chemin passant par toutes les cases
- Respecter sens/direction de chaque flèche



Sujet 11 — Signpost : analyse théorique et génération contrainte de grilles

Signpost : jeu type Puzzle

- Faire chemin passant par toutes les cases
- Respecter sens/direction de chaque flèche



Sujet 11 — Signpost : analyse théorique et génération contrainte de grilles

Signpost : jeu type Puzzle

- Faire chemin passant par toutes les cases
- Respecter sens/direction de chaque flèche

1 ↓	b+3 ↓	a+2 ↓
b+1 →	b+2 ↑	a+1 ↑
b • ↑	a • ↗	9 • ★

Sujet 11 — Signpost : analyse théorique et génération contrainte de grilles

Signpost : jeu type Puzzle

- Faire chemin passant par toutes les cases
- Respecter sens/direction de chaque flèche

1 ↓	b+3 ↓	b+6 ↓
b+1 →	b+2 ↑	b+5 ↑
b • ↑	b+4 ↗	9 • ★

Sujet 11 — Signpost : analyse théorique et génération contrainte de grilles

Signpost : jeu type Puzzle

- Faire chemin passant par toutes les cases
- Respecter sens/direction de chaque flèche

1 ↓	5 ↓	8 ↓
3 →	4 ↑	7 ↑
2 ↑	6 ↗	9 ★

Sujet 11 — Signpost : analyse théorique et génération contrainte de grilles

Signpost : jeu type Puzzle

- Faire chemin passant par toutes les cases
- Respecter sens/direction de chaque flèche

1 ↓	5 ↓	8 ↓
3 →	4 ↑	7 ↑
2 ↑	6 ↗	9 ★

Buts

- Améliorer génération grille dans code existant (C)
 - Ajout contraintes
 - Niveaux de difficulté pour un humain ?
- Analyser théoriquement le jeu
 - Unicité des solutions ?
 - Difficulté du jeu ?
 - ...