

5. Gestion des données et vie privée du citoyen urbain

- Architectures de gestion de données face au respect de la vie privée
- Gestion de la vie privée dans les réseaux sociaux mobiles
- **Privacy-by-design: gestion de données confinées (puces et capteurs)**
- Gestion de la vie privée dans les applications mobiles participatives
- Traitements de données globaux respectueux de la vie privée

Architectures Privacy-by-Design pour les villes

- **Ce que ça n'est pas ?**
 - Une architecture pour cacher des secrets

Architectures Privacy-by-Design pour les villes

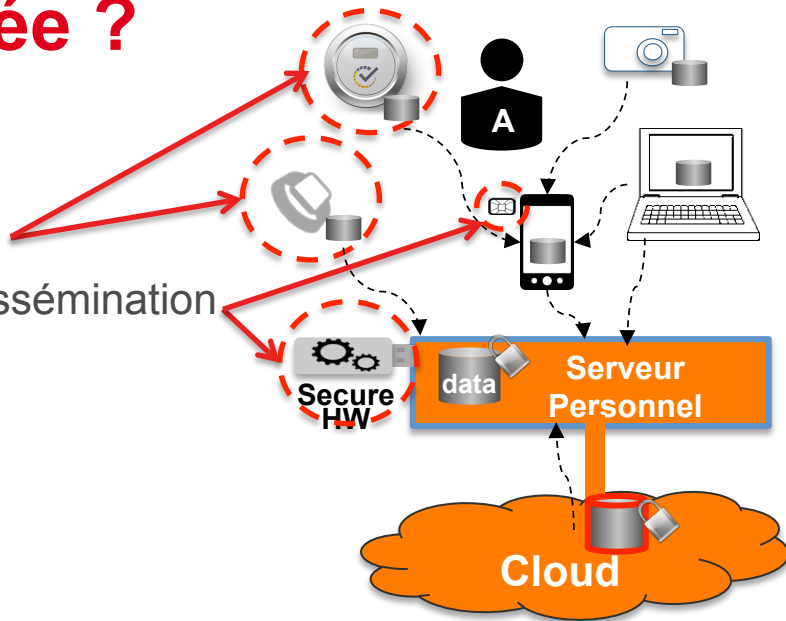
- Ce que ça n'est pas ?
 - Une architecture pour cacher des secrets
- **Ce que c'est ?**
 - Permettre à l'individu de contrôler la collecte, l'usage et le partage de ses données
 - Proscrire par construction les usages secondaires « cachés » réalisés sans consentement

Architectures Privacy-by-Design pour les villes

- Ce que ça n'est pas ?
 - Une architecture pour cacher des secrets
- Ce que c'est ?
 - Permettre à l'individu de contrôler la collecte, l'usage et le partage de ses données
 - Proscrire par construction les usages secondaires « cachés » réalisés sans consentement
- **Comment faire ?**
 - Privacy-by-Design
 - Protection du cycle de vies des données de bout en bout
 - Réguler la dissémination des données depuis les sources
... jusqu'au cloud

Pourquoi l'évaluation confinée ?

- P-b-D: réguler le cycle de vie des données
 - De petits objets assurent une part de la collecte
 - Des composants sécurisés peuvent réguler la dissémination

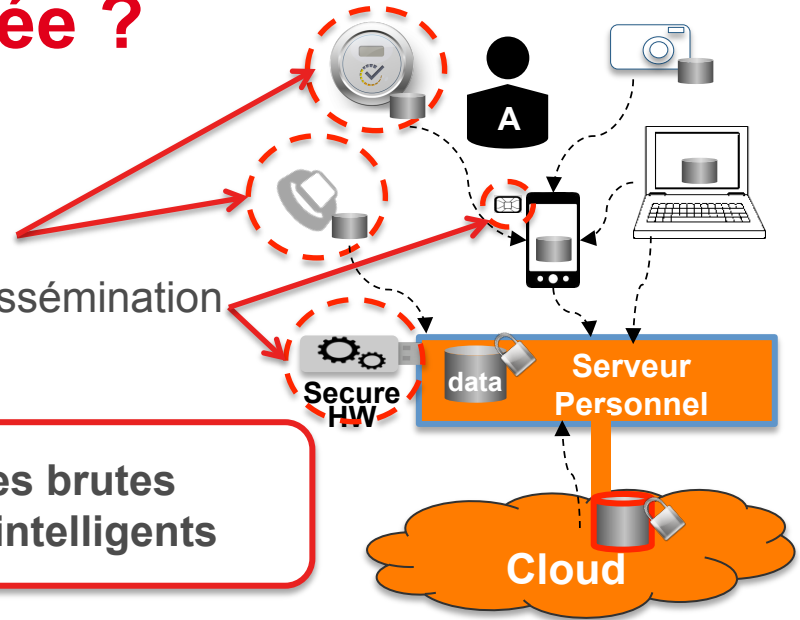


Pourquoi l'évaluation confinée ?

- P-b-D: réguler le cycle de vie des données
 - De petits objets assurent une part de la collecte
 - Des composants sécurisés peuvent réguler la dissémination

Disséminer des résultats et non les données brutes
Gestion de données embarquée pour objets intelligents

- Techniques de gestion de données pour petits objets
 - Pour ne pas surcharger les serveurs / l'infrastructure



Gestion de données : une architecture commune

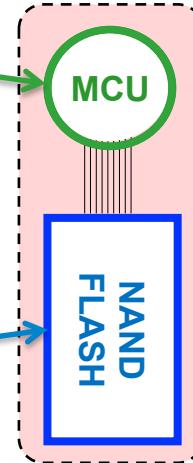
(capteurs à mémoire, token/clé USB/carte SD sécurisés, carte SIM multimédia, IoT...)

Microcontrôleur

- Faible coût
- Faible consommation
- Protection contre les attaques physiques
 - ✓ Miniaturisation, composants superposés,
 - ✓ Maillage (signal porteur), Capteurs (lum./freq./volt.)

Mémoire FLASH

- Faible coût
- Faible consommation
- Densité et robustesse



Gestion de données : une architecture commune

(capteurs à mémoire, token/clé USB/carte SD sécurisés, carte SIM multimédia, IoT...)

Microcontrôleur

- Faible coût
- Faible consommation
- Protection contre les attaques physiques
 - ✓ Miniaturisation, composants superposés,
 - ✓ Maillage (signal porteur), Capteurs (lum./freq./volt.)

Très petite RAM
(<128 KB)

Mémoire FLASH

- Faible coût
- Faible consommation
- Densité et robustesse

Ecritures aléatoires coûteuses

Time (ms) per I/O	Read	Seq. Write	Rand. Write
Kingston µSDHC	1.3	6.3	160
Lexar SDMI4GB-715	0.8	2.1	180
Samsung µSDHC Plus	1.3	2.9	315
SiliconPower SDHC	1.4	3.4	40

Gestion de données : une architecture commune

(capteurs à mémoire, token/clé USB/carte SD sécurisés, carte SIM multimédia, IoT...)

Microcontrôleur

- Faible coût
- Faible consommation
- Protection
 - ✓ Miniaturisation
 - ✓ Maillage

Très petite RAM
(<128 KB)



**Contraintes fortes et durables
inhérentes aux dispositifs autonomes,
à bas coût (capteur, carte SD),
ou sécurisés (puce sécurisée ou “durcie”)**

Mémoire FLASH

- Faible coût
- Faible consommation
- Densité et robustesse

Ecritures aléatoires coûteuses

Time (ms) per I/O	Read	Seq. Write	Rand. Write
Kingston µSDHC	1.3	6.3	160
Lexar SDMI4GB-715	0.8	2.1	180
Samsung µSDHC Plus	1.3	2.9	315
SiliconPower SDHC	1.4	3.4	40

Formulation du problème

- **Objectif** : réaliser des traitements de données (requêtes) avec peu de RAM sur de grandes quantités de données stockées en FLASH NAND

**Evaluer des requêtes
avec une petite RAM**

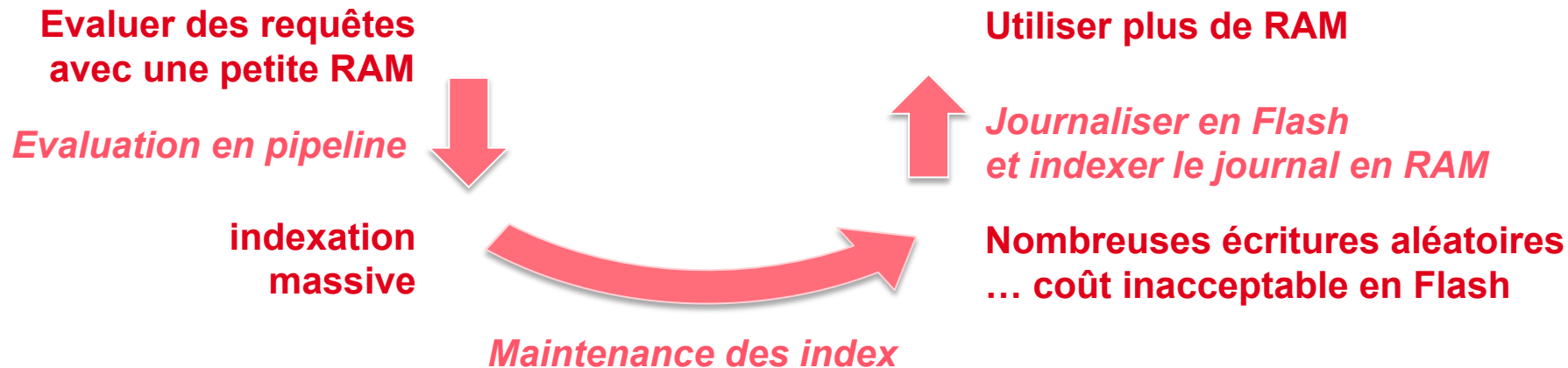
Evaluation en pipeline



**indexation
massive**

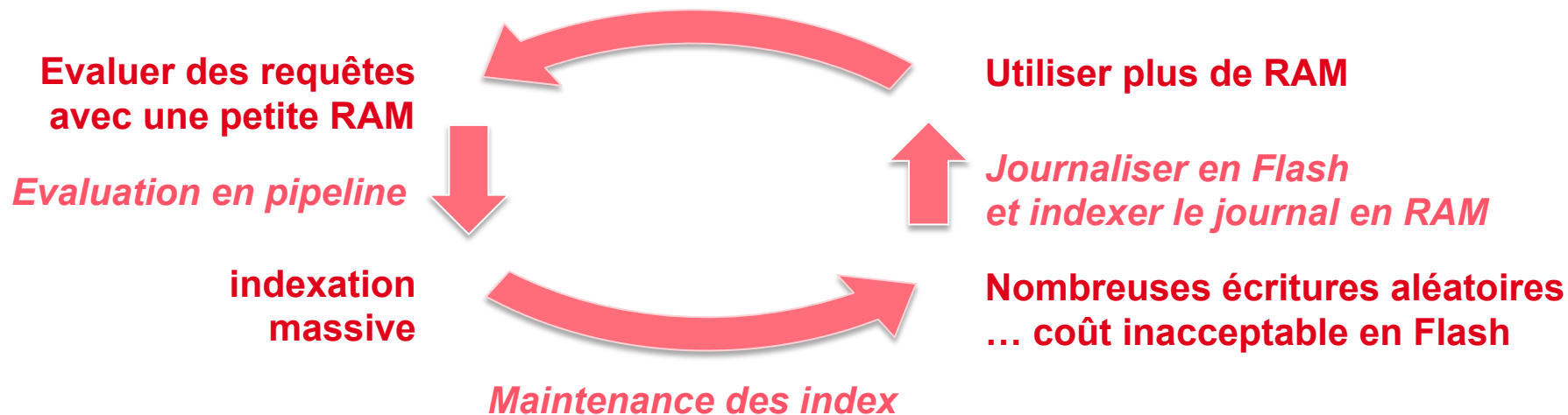
Formulation du problème

- **Objectif** : réaliser des traitements de données (requêtes) avec peu de RAM sur de grandes quantités de données stockées en FLASH NAND



Formulation du problème

- **Objectif** : réaliser des traitements de données (requêtes) avec peu de RAM sur de grandes quantités de données stockées en FLASH NAND



Comment rompre ce cercle vicieux ?

Stratégie générale pour résoudre ce problème

1. Concevoir des index permettant une évaluation en pipeline

Stratégie générale pour résoudre ce problème

1. Concevoir des index permettant une évaluation en pipeline
2. **Organiser ces index dans des structures séquentielles**
 - ... qui satisfont par construction les contraintes de la Flash**
 - Ecrire séquentiellement dans chaque structure (et ne jamais le modifier)
 - de manière à proscrire les écritures aléatoires par construction
 - Allouer & libérer des zones de grande taille (multiple d'un bloc de Flash)
 - pour éviter l'émiettement de la mémoire (ramasse miettes coûteux)

Stratégie générale pour résoudre ce problème

1. Concevoir des index permettant une évaluation en pipeline
2. Organiser ces index dans des structures séquentielles
3. **Passer à l'échelle par réorganisations successives**
 - Toutes les structures séquentielles sont réécrites et transformées
... l'opération de transformation repose sur des structures séquentielles uniquement

Stratégie générale pour résoudre ce problème

1. Concevoir des index permettant une évaluation en pipeline
2. Organiser ces index dans des structures séquentielles
3. Passer à l'échelle par réorganisations successives

De nombreuses techniques sont supportées aujourd'hui en embarqué. Des résultats peuvent être calculés à l'intérieur du dispositif, sans devoir externaliser toutes les données collectées.

Conclusion

- **Nombre croissant de fonctionnalités embarquées support au P-b-D**
 - Bases de données : Antelope [Sensys11], MiloDB [DAPD14], LittleD [SAC14]
 - Système clé valeurs : IonDB [CCECE15]
 - Recherche textuelle : Microsearch [Tan10], SSF [Anciaux15]
 - Reconnaissance d'images [Yan08]
 - Gestion de traces de géolocalisation [Ton15]
- **De nombreux défis restent ouverts**
 - Séries temporelles, RDF, NoSQL
 - Approche générale de co-conception
 - ✓ Calibrer l'architecture et les flots de données en fonction des traitements P-b-D réalisables

Références

- [Cavoukian11] Cavoukian, A. (2011). “Privacy by Design - The 7 Foundational Principles.” <http://www.ipc.on.ca/images/Resources/7foundationalprinciples.pdf>
- [Cavoukian12] Cavoukian, A., & Jonas, J. (2012). *Privacy by design in the age of big data*. Information and Privacy Commissioner of Ontario, Canada.
- [PlugDB] <https://project.inria.fr/plugdb/>
- [Tan10] Tan, C. C., Sheng, B., Wang, H., & Li, Q. (2010). Microsearch: A search engine for embedded devices used in pervasive computing. *ACM Transactions on Embedded Computing Systems (TECS)*, 9(4).
- [Anciaux15] Anciaux, N., Lallali, S., Popa, I. S., Pucheral, P. A Scalable Search Engine for Mass Storage Smart Objects. *PVLDB* 8(9), 2015.
- [Anciaux14] Anciaux, N., Bouganim, L., Pucheral, P., Guo, Y., Le Folgoc, L. MiloDB: a Personal, Secure and Portable Database Machine. *Distributed and Parallel Databases (DAPD)*, Vol. 32(1), 2014.
- [Ton15] Ton That, D. H., Popa, I. S., Zeitouni, K. TRIFL: A Generic Trajectory Index for Flash Storage. *ACM Transactions on Spatial Algorithms and Systems* 1(2), 2015.
- [Yan08] Yan, T., Ganesan, D., and Manmatha, R. Distributed image search in camera sensor networks. In *Proc. of the 6th ACM Conference on Embedded Network Sensor Systems, SenSys'08*, pp. 155–168, 2008.
- [Douglas14] Douglas, G., & Lawrence, R. (2014, March). LittleD: a SQL database for sensor nodes and embedded applications. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing* (pp. 827-832). ACM.
- [CCECE15] Fazackerley, S., Huang, E., Douglas, G., Kudlac, R., & Lawrence, R. (2015, May). Key-value store implementations for Arduino microcontrollers. In *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on* (pp. 158-164). IEEE. IonDB, Link: https://www.reddit.com/r/arduino/comments/34uu8r/iondb_is_a_key_value_store_for_the_arduino/