ENSTA BRETAGNE

Lab-STICC UMR 6285

Storage in the Cloud, data placement issues and some solutions

Jalil Boukhobza,
ENSTA-Bretagne, Lab-STICC UMR 6285

Who am I ?

https://www.ensta-bretagne.fr/boukhobza/
jalil.boukhobza@ensta-bretagne.fr

- **Education**
  - 1999 – Engineer in Electronics, INELEC, Algeria
  - 2000 – Master (DEA) in computer science, Univ. Versailles
  - 2004 – PhD Univ. Versailles, PRiSM Lab., Storage Systems
- **Prof. Exp.**
  - 2004-2006 – Research and teaching assistant, Univ. Versailles
  - 2006-2020 – Associate Professor, Univ. Bretagne Occidentale
  - 2013-* – Part time researcher at IRT b<>com, Rennes
  - 2016 Invited researcher, Hong Kong Polytechnic University
  - 2020-* Professor, ENSTA-Bretagne / team leader of SHAKER (Software/HArdware and unKnown Environment inteRactions)

- **Research topics:**
  - Storage and memory systems
    - Modeling / benchmarking / data placement / I/O optimization / I/O tracing
  - Domains: Cloud and Fog resource management, Embedded systems, HPC, DB
  - Current projects:
    - CEA: DataMeSS, data placement in multi-tiered storage systems
    - Atos: Energy I/O optimization for HPC with frugal and federated learning
    - NIST: cache optimization for NDN networks
    - DGA-AID project: DISPEED Intrusion Detection and Security/Performance/Energy tradeoff: a Study for Drone Swarms, with UBO, NAval Group, ICS FORTH, Scientific coordinato
    - IRT b<>com: service scheduling in heterogeneous systems (FPGA, GPU, GPP)

Disclaimer

- The focus of this presentation is about data placement on the Cloud
- Most SOTA work on the topic have been omitted as we focus on our contributions → see papers for SOTA work
- Simplification = loss of information

- I am not
  - An Operations Research (OR) expert
  - An Artificial Intelligence (AI) expert
  - But I use those concepts

- My vision of things is necessariliy biased
  - Computer scientist
  - Working on system research
  - Working in academia
  - not a « technology » guy ☺

- Assumption about the audience:
  - Mainly computer scientists
  - Working in a Cloud/edge topic
  - Unaware of storage systems intricacies

- 2 types of slides
  - Normal slides
  - Quick slides →

I will frequently say:

jalil.boukhobza@ensta-bretagne.fr

Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

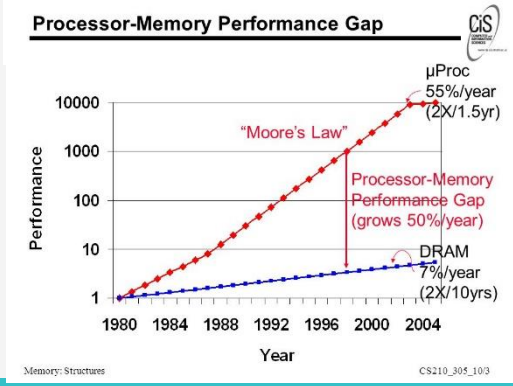
« It's the Memory, Stupid! »

A paper written by Richard Sites (1996), *lead designer* at DEC

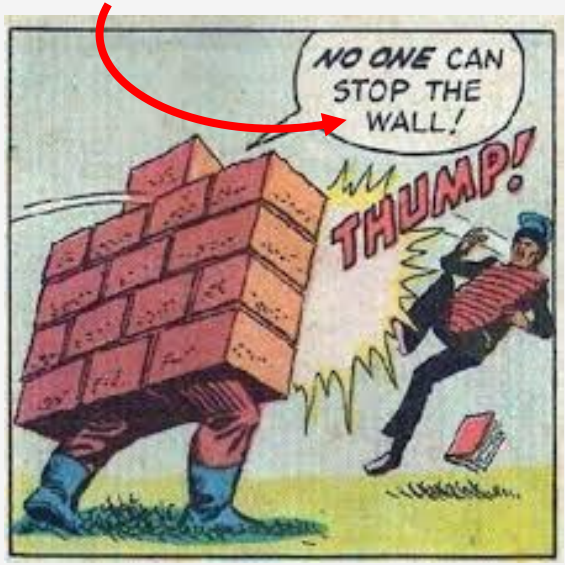
"Across the industry, today's chips are largely able to execute code faster than we can feed them with instructions and data. There are no longer performance bottlenecks in the floating-point multiplier or in having only a single integer unit. The real design action is in memory subsystems— caches, buses, bandwidth, and latency."

"Over the coming decade, memory subsystem design will be the only important design issue for microprocessors."

– Richard Sites, after his article "It's The Memory, Stupid!", Microprocessor Report, 10(10),1996



Memory ☺



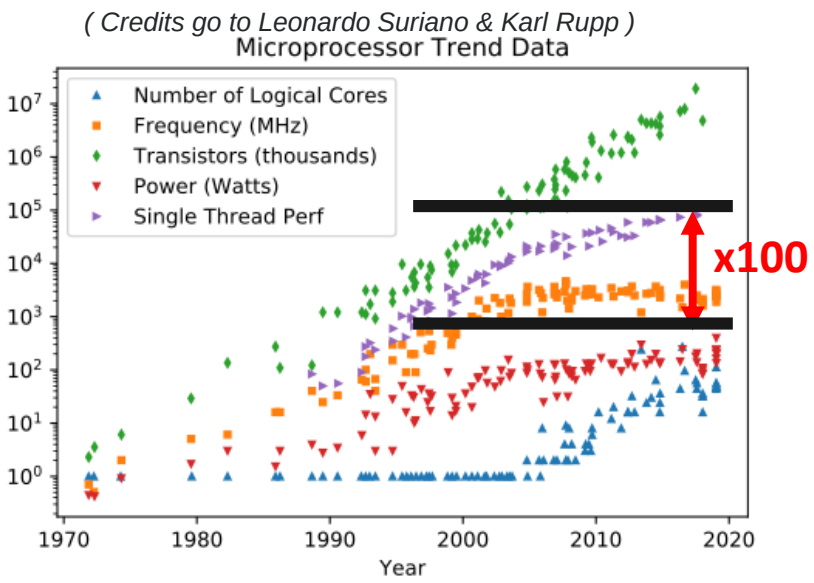
Source: Spidey ed. Lug

$$t_{avg} = p \times t_c + (1-p) \times t_m$$

Wulf_94

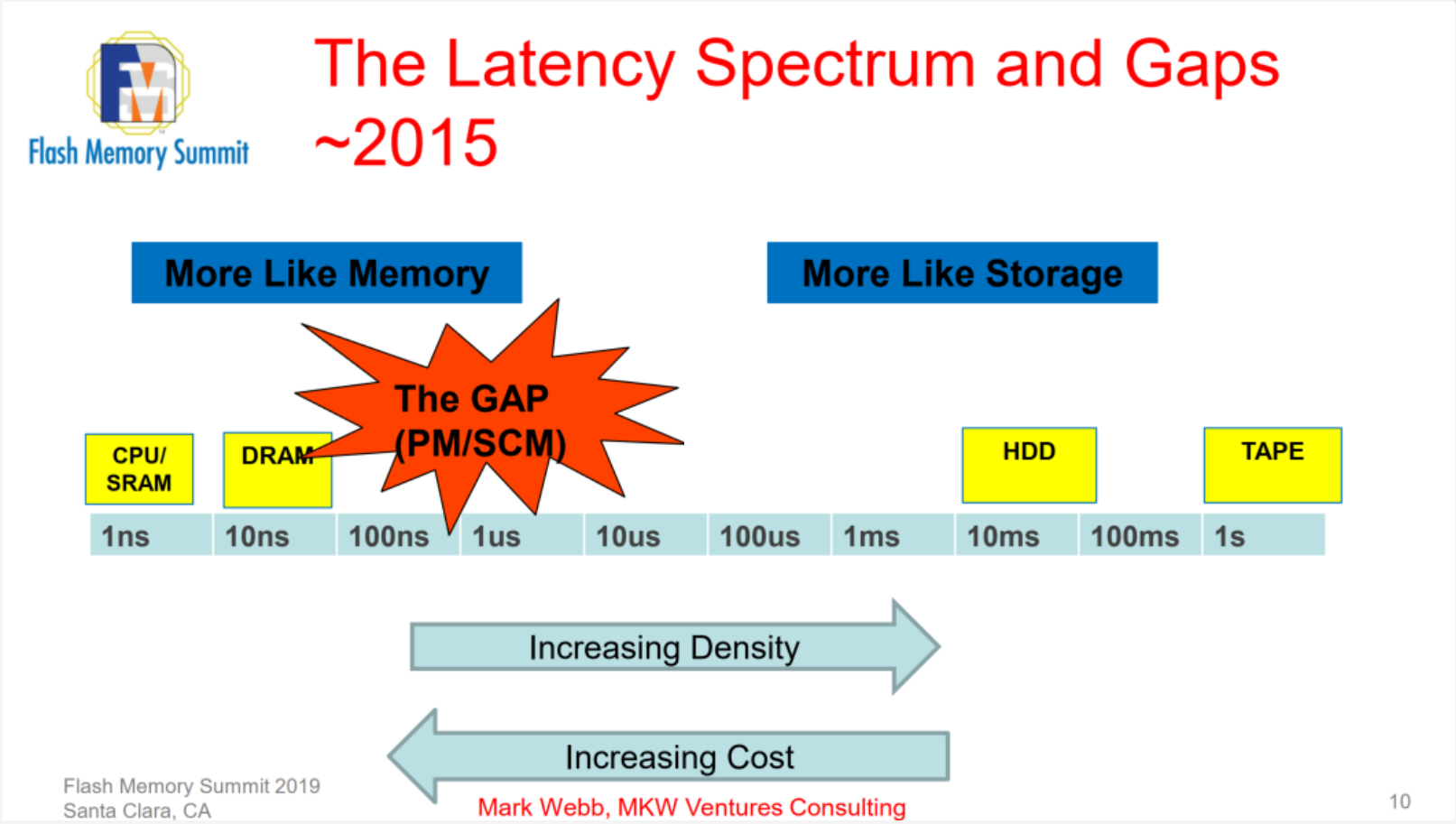
Now, although $(1-p)$ is small, it isn't zero. Therefore as t_c and t_m diverge, t_{avg} will grow and system performance will degrade. In fact, it will hit a wall.

& Latency

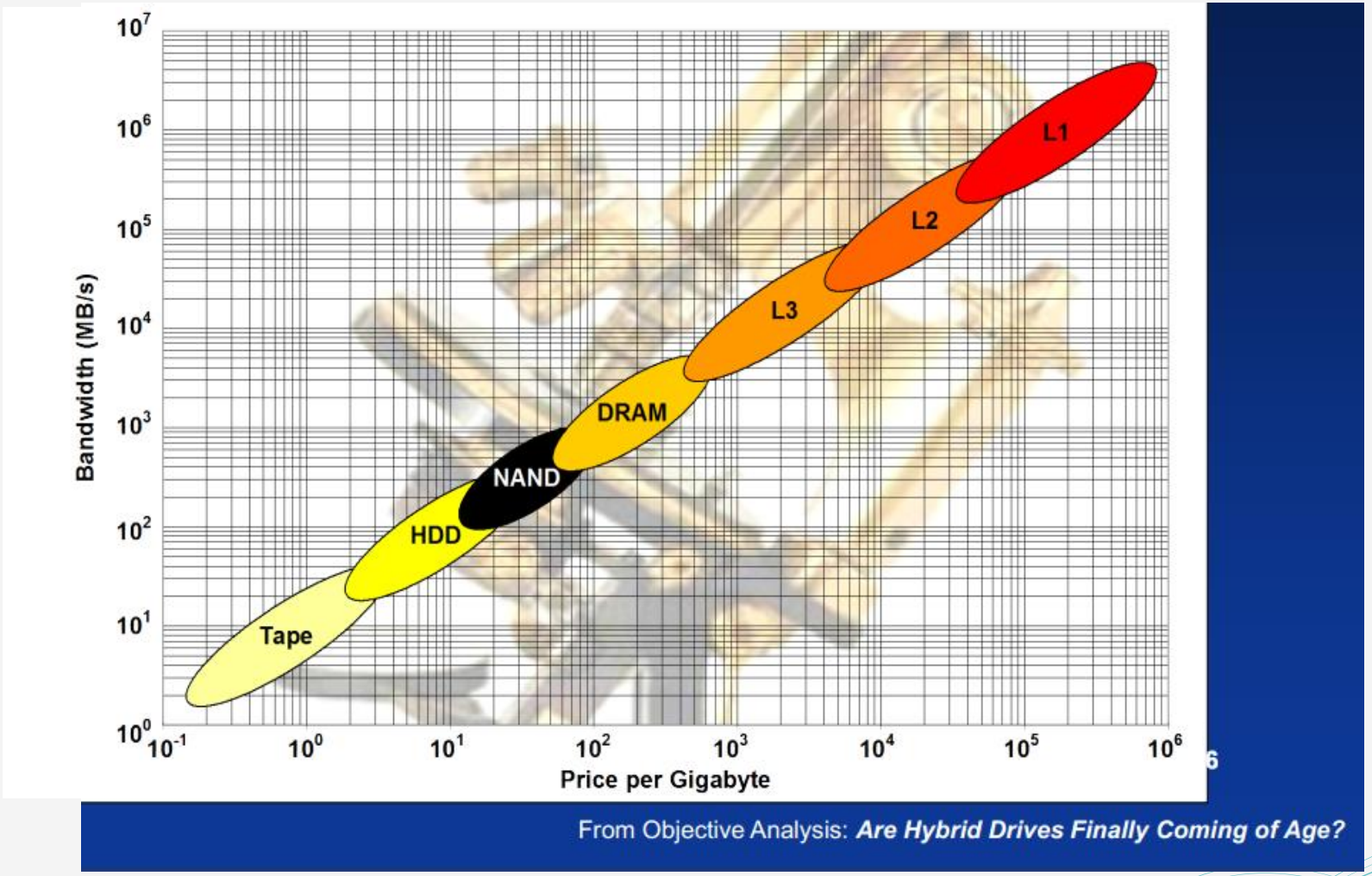


Slide of O. Mutlu
https://safari.ethz.ch/memory_systems/TUWien2019/doku.php?id=schedule

The gap with storage is even worse…

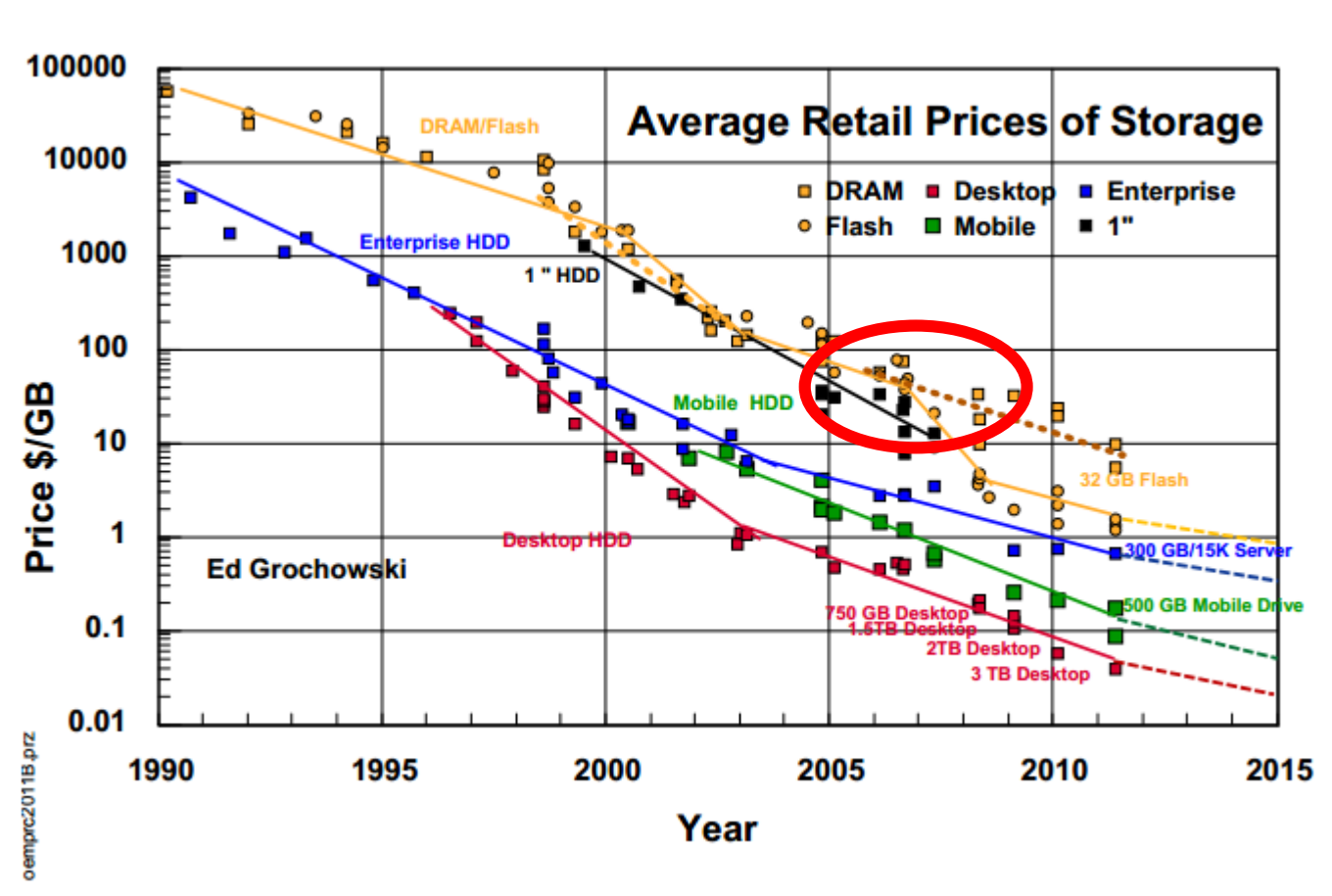


Context – ideal (memory) world



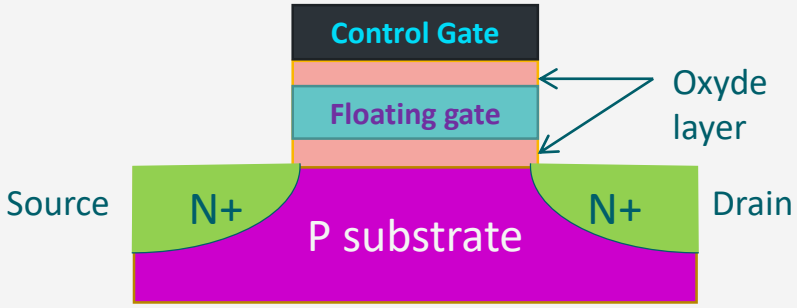
jalil.boukhobza@ensta-bretagne.fr

Price of memory



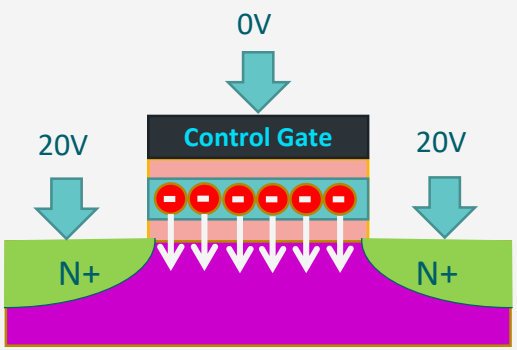
Flash memory cells

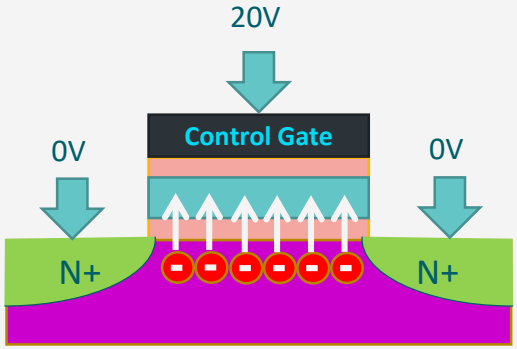
- Invented by F. Masuoka Toshiba 1980
- Introduced by Intel in 1988
- Type of EEPROM (Electrically Erasable & Programmable Read Only Memory)
- Use of Floating gate transistors
- Electrons pushed in the floating gate are trapped

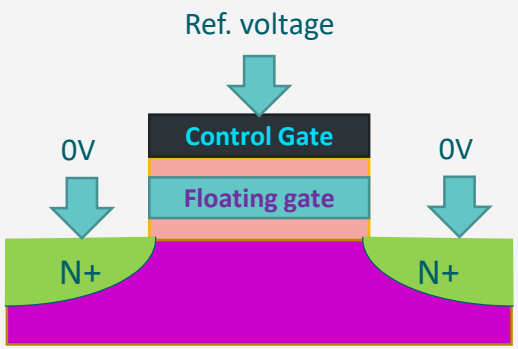


- 3 operations: program (write), erase, and read

Flash memory operations







Erase operation

- FN (Fowler-Nordheim) tunneling: Apply high voltage to substrate (compared to the operating voltage of the chip - usually between 7– 20V)
- → electrons off the floating gate
- Logic « 1 » in SLC

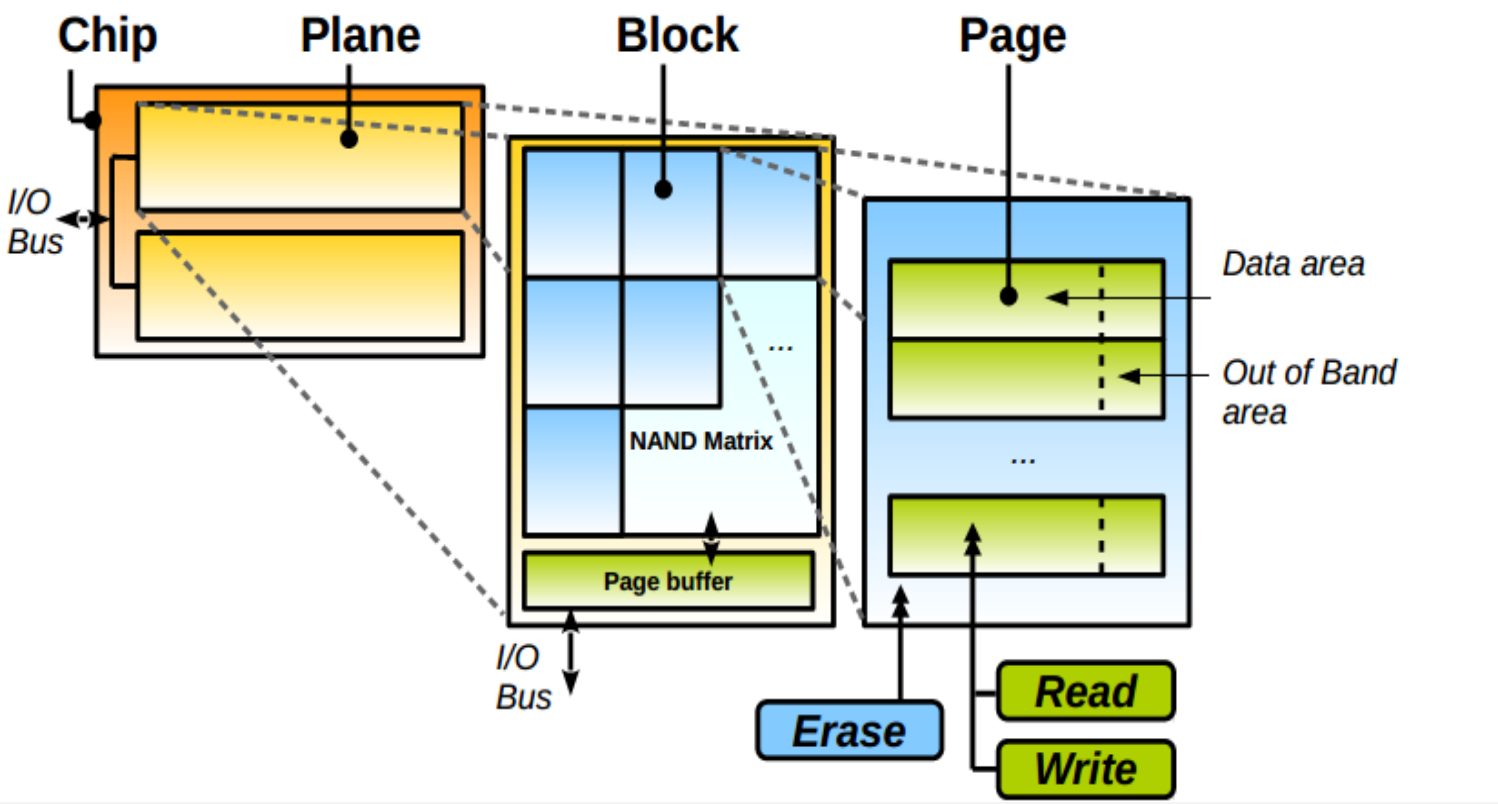
Program / write operation

- Apply high voltage to the control gate
- → electrons get trapped into the floating gate
- Logic « 0 »

Read operation

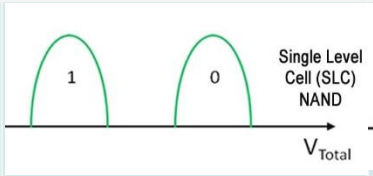
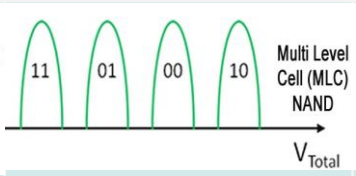
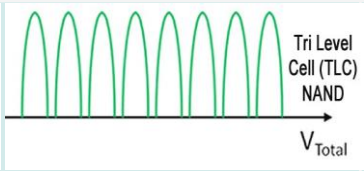
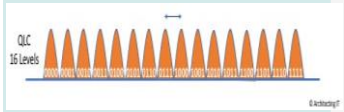
- Apply reference voltage to the control gate:
  - If floating gate charged: no current flow
  - If not charged; current flow

NAND flash memory architecture

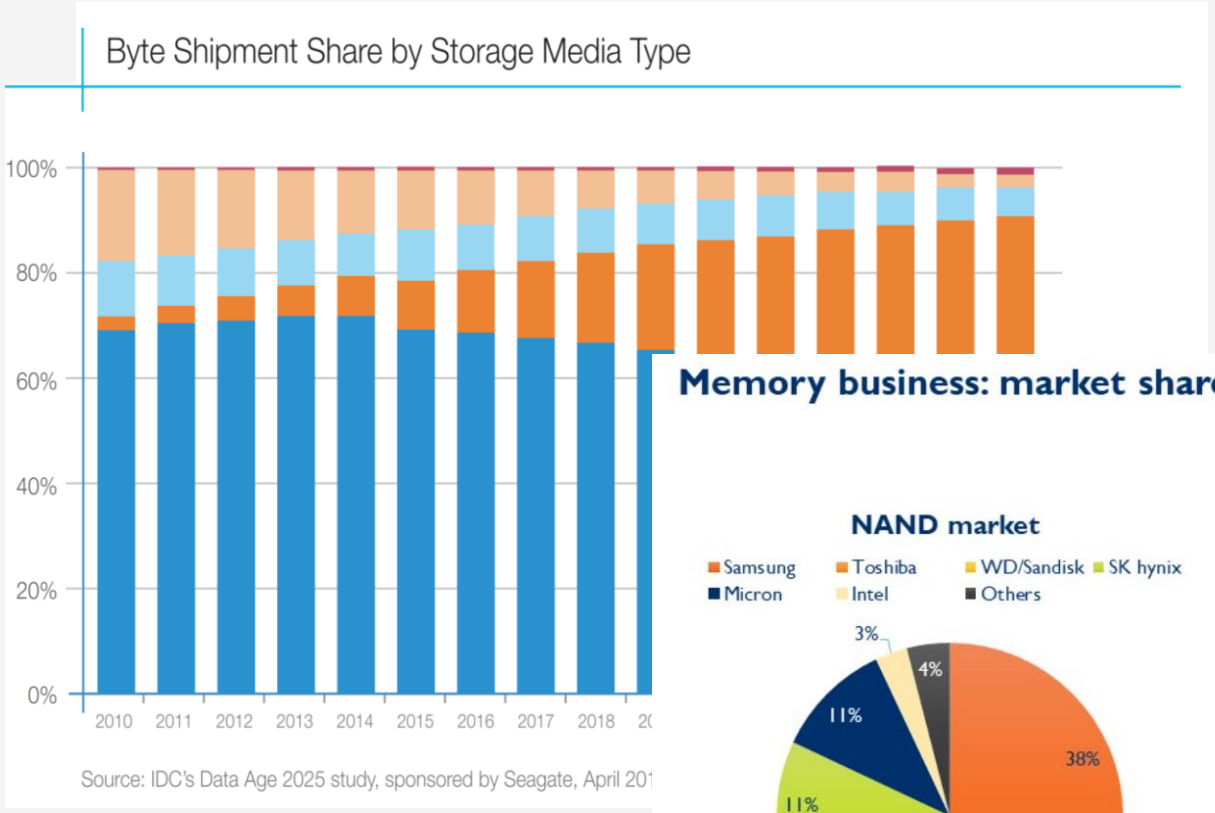


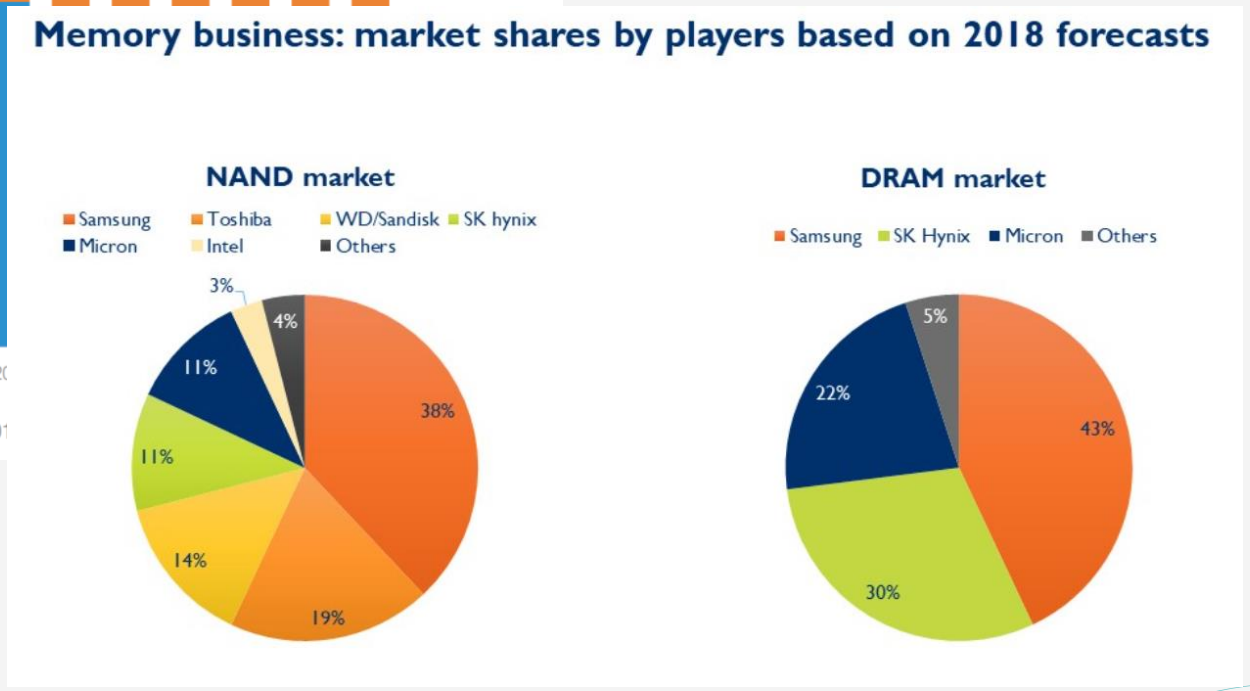
- Read/Write → page
- Erasures → blocks
- Page: 2-8KB
- Block: 128-4096 KB

Different densities: SLC, MLC, TLC, QLC

	SLC (Single Level Cell)	MLC (Multi Level Cell)	TLC (Tri Level Cell)	QLC (Quad Level Cell)
				
Storage	1 bit / cell	2 bits / cell	3 bits /cell	4 bits/cell
Performance	++++	+++	++	+
Density	+	++	+++	++++
Lifetime (P/E cycles)	~ 100 000	~ 10 000	~5 000	~1000
ECC complexity	+	++	+++	++++
Applications	Embedded and industrial applications (high end SSDs…)	Most consumer applications (e.g. memory cards)	Low-end consumer applications not needing data updates (e.g. mobile GPS)	Write once, read many

Byte shipment share





Flash memory constraints

Write/Erase granumarity asymmetry (**Cons1**)

Erase-before-write rule (**Cons2**)

Limited cell lifetime (**Cons3**)

+

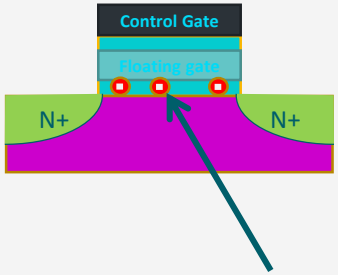
Data update:
- Invalidate
- Out-of-place update

+

Logical to physical mapping

Garbage Collection

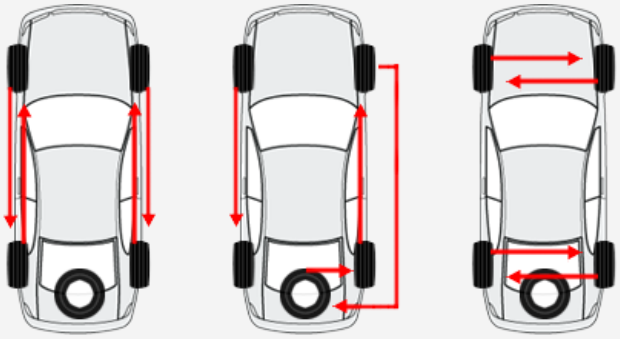
Wear leveling



- Electrons get trapped in the oxide layer deteriorating its characteristics
- Electrons cannot move from oxyde layer to floating gate

Wear leveling

/ You already do that with your tyres …

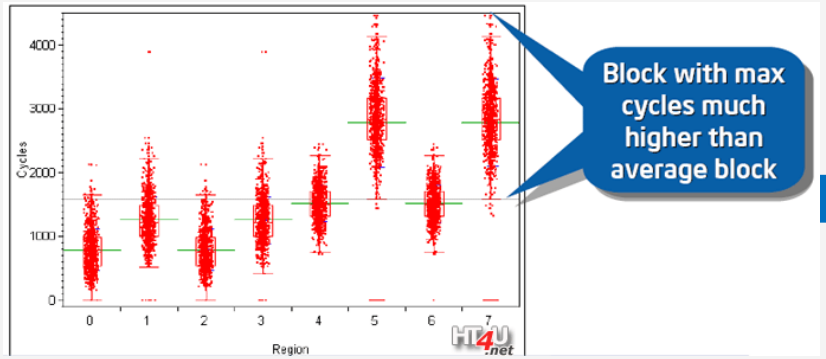


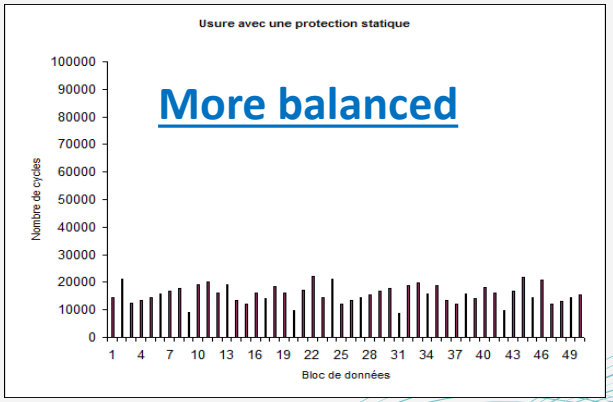
4 tyre rotation all are same size | 5 tyre rotation all are same size | Tyre rotation when size is different front & rear

Pierelli Courtesy

/ Keeping a balanced erasures' distribution over flash memory blocks.

http://www.presence-pc.com/tests/ssd-flash-disques-22675/5/





Garbage Collection

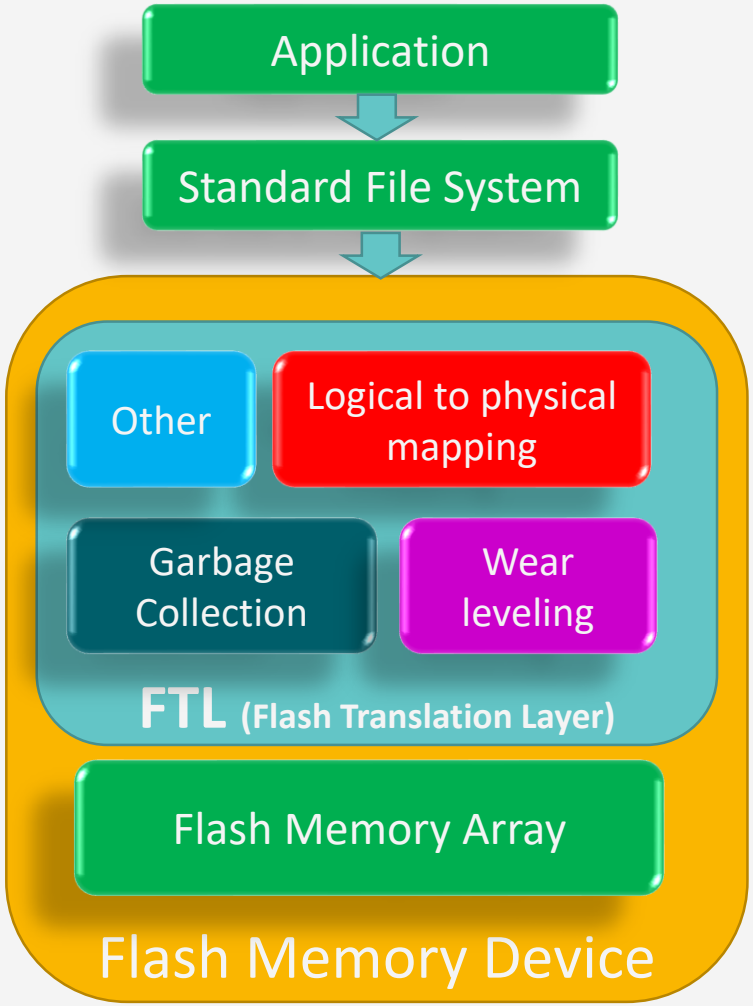
Inv. Op.

Moving people to a new city and « erasing » the old one to reuse the space !!!

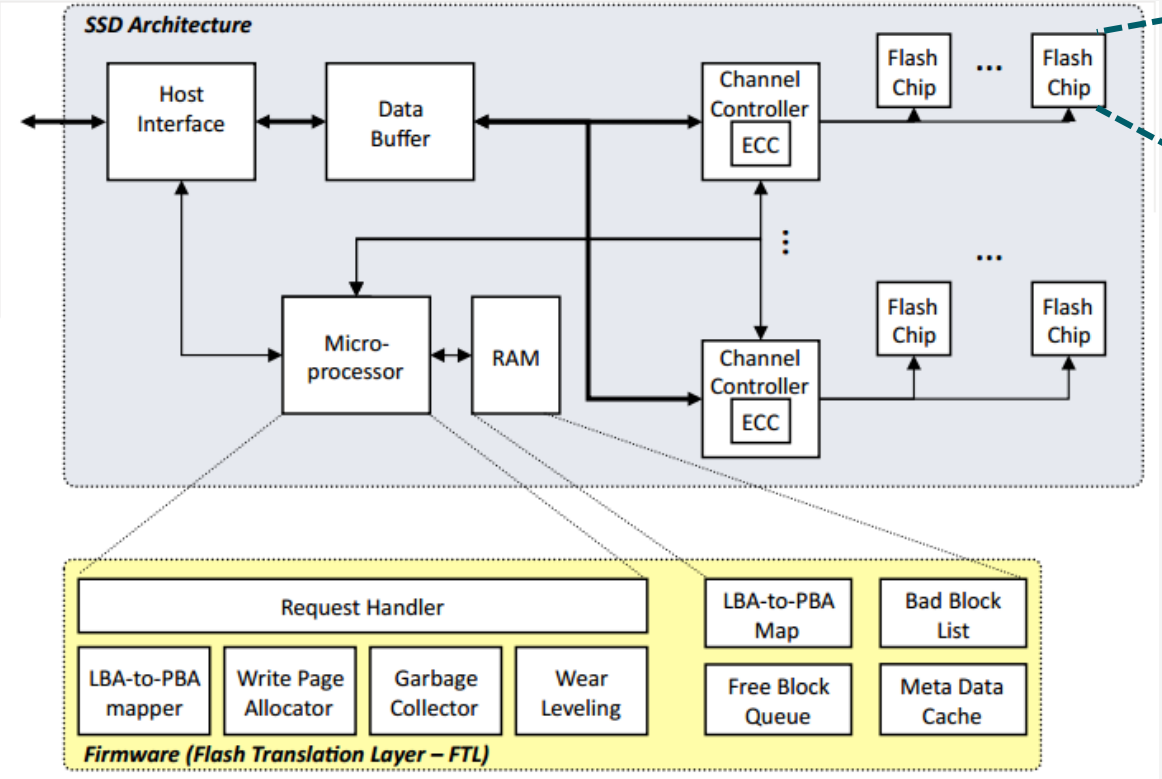
/ Moving valid pages from blocks containing invalid data and then erase/recycle the blocks

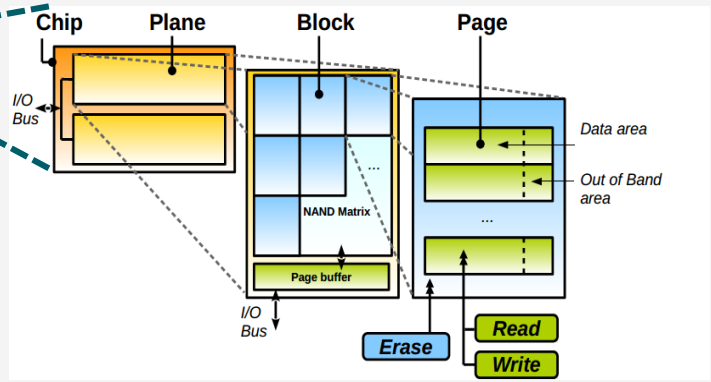
jalil.boukhobza@ensta-bretagne.fr

Flash memory structure



SSD architecture

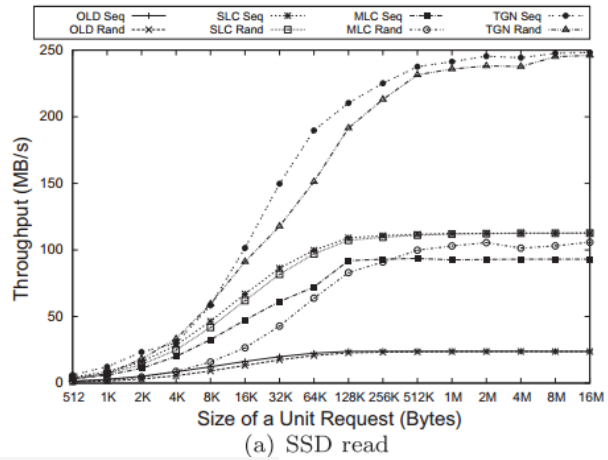




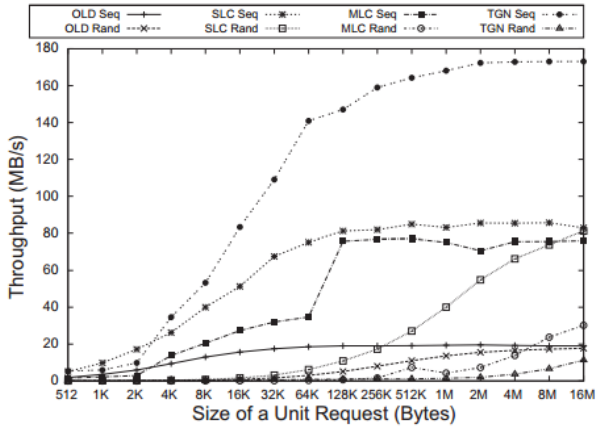
Source: Bonnet, Bouganim, Koltsidas, Viglas, VLDB 2011

Read/write asymmetry and disparity

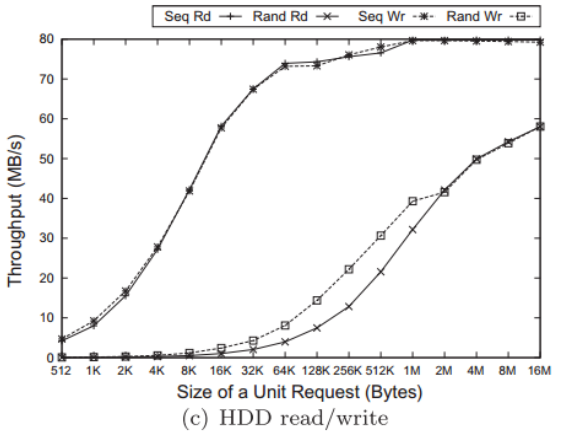
- **Flash disk performance is heterogeneous**
  - Depend on internal structure and workload
  - Performance disparities between SSDs from the same constructor and between different technologies are significantly high



(a) SSD read



(b) SSD write



(c) HDD read/write

Table 3
Specifications of the storage devices used in our work.

	OLD	MLC	SLC	TGN	HDD
Model	FSD32GB25M	1C32G	MSP7000	MMCRE28G5	WD1600BEKT
Vendor	Super talent	OCZ	MTron	Samsung	Western digital
Form factor	2.5 in.	2.5 in.	2.5 in.	2.5 in.	2.5 in.
Flash type/RPM	SLC	MLC	SLC	MLC	7200
Capacity	32 GB	32 GB	16 GB	128 GB	160 GB
Rd./Wr. Perf. (MB/s)	60/45	143/93	120/90	220/200	NA/NA

Performance of write operations

Flash performance needs time to reach steady state… and may oscillate between states …

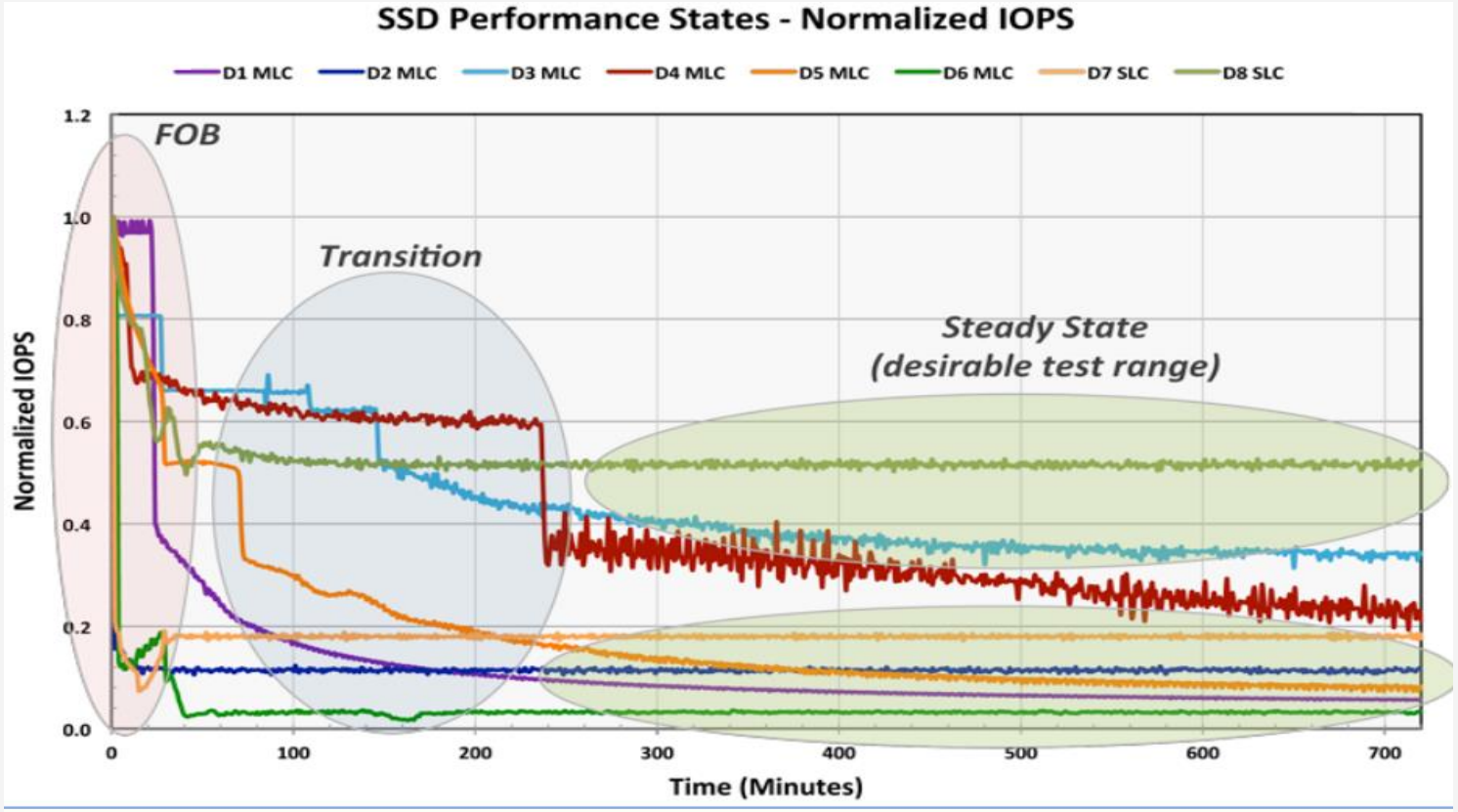


Figure 1-1 – NAND-based SSS Performance States (RND 4KiB Writes)

Source: http://snia.org/sites/default/files/SSS%20PTS%20Client%20-%20v1.1.pdf

Summary

As compared to traditional drives

- **New constraints**
  - Write/Erase granularity
  - Erase before write
  - Endurance
- **New performance model**
  - ~ Symmetric sequential / random read
  - More sensitive to writes (lifetime)
    - Asymmetric sequential / random writes
  - Sensitive to the fill rate

Architecture

- Cache
- FTL

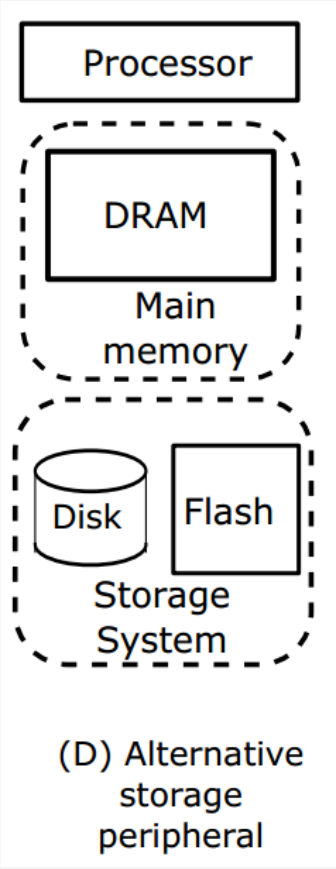
jalil.boukhobza@ensta-bretagne.fr

Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

Hybrid storage for the Cloud

/ Many flash memory integration options:

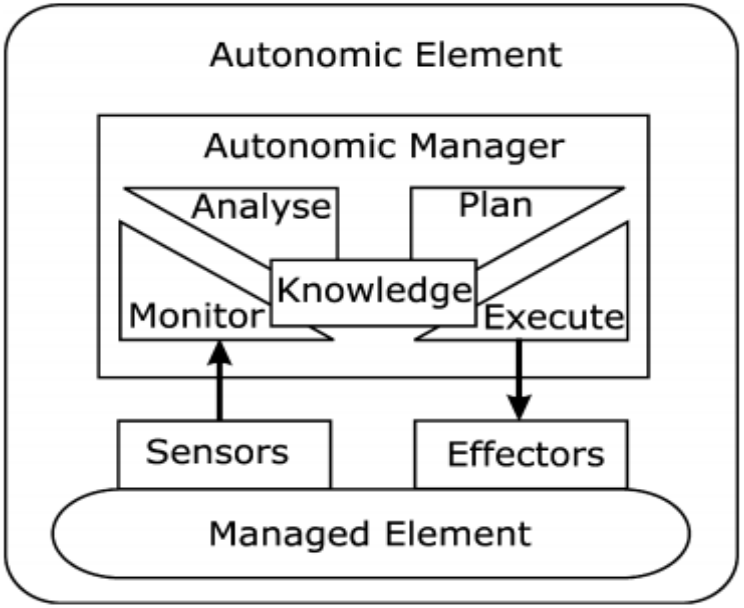


(D) Alternative storage peripheral

J. Boukhobza, "**Flashing in the Cloud: shedding some light on NAND flash memory storage systems**", in book Data Intensive Storage Services for Cloud Environments, , pp. 241-266, IGI Global Editor, ISBN13: 9781466639348, Apr. 2013

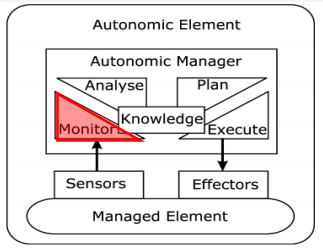
MAPE-K model

- MAPE-K [IBM01] autonomic loop (for data placement)
- Managed Element (ME)
  - Any software or hardware element
- Sensors
  - Collect the data about ME
- Effectors
  - Carry out changes to ME
- Autonomic Manager (AM)
  - **Monitor** ME and Execute changes
  - **Analyze** the monitored data
  - **Plan** for changes if requires
  - **Execute** planned changes
  - **Knowledge** : refers to the information that AM may maintain about ME

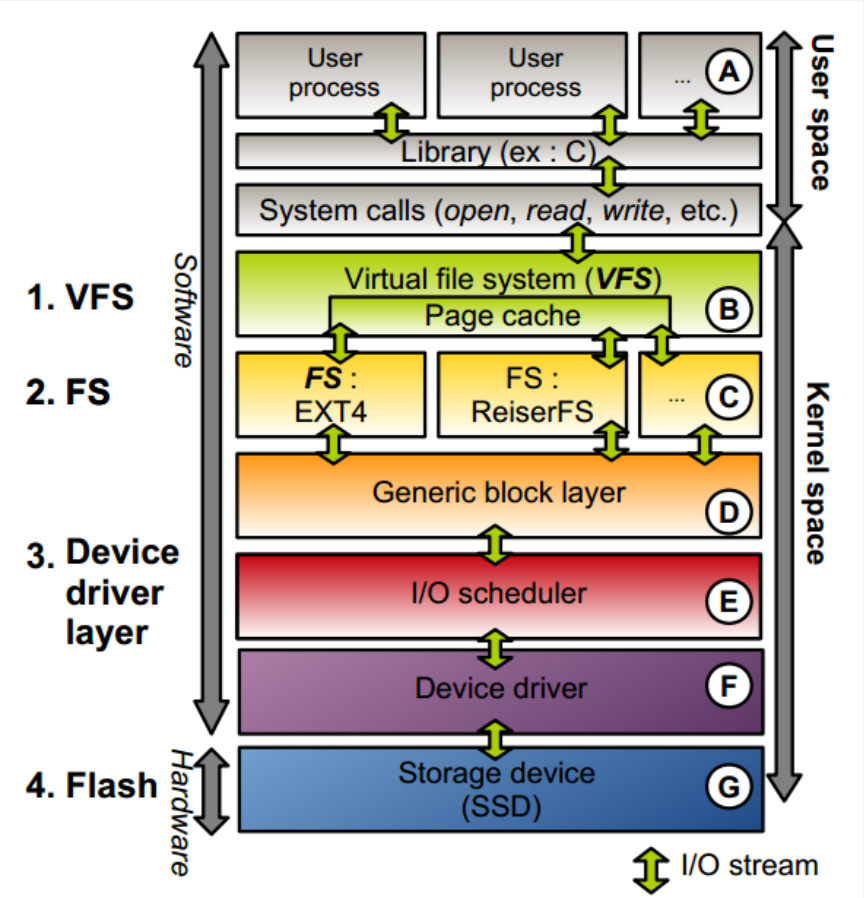


[IBM01] Jeffrey O Kephart and DavidMChess. The vision of autonomic computing. Computer, 36(1) :41–50, 2003.

1) Monitoring I/Os – intro

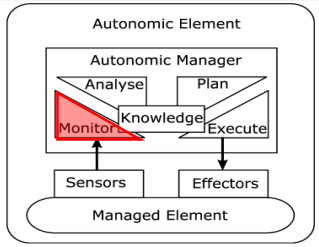


- An I/O request passes through a large software stack
  - Libraries: buffering (e.g.fflush())
  - VFS : file semantic
  - FS : file structure and hierarchy
  - I/O scheduler: optimize I/O by reordering I/Os
  - Storage device: yet several other blocks (see previous slides)

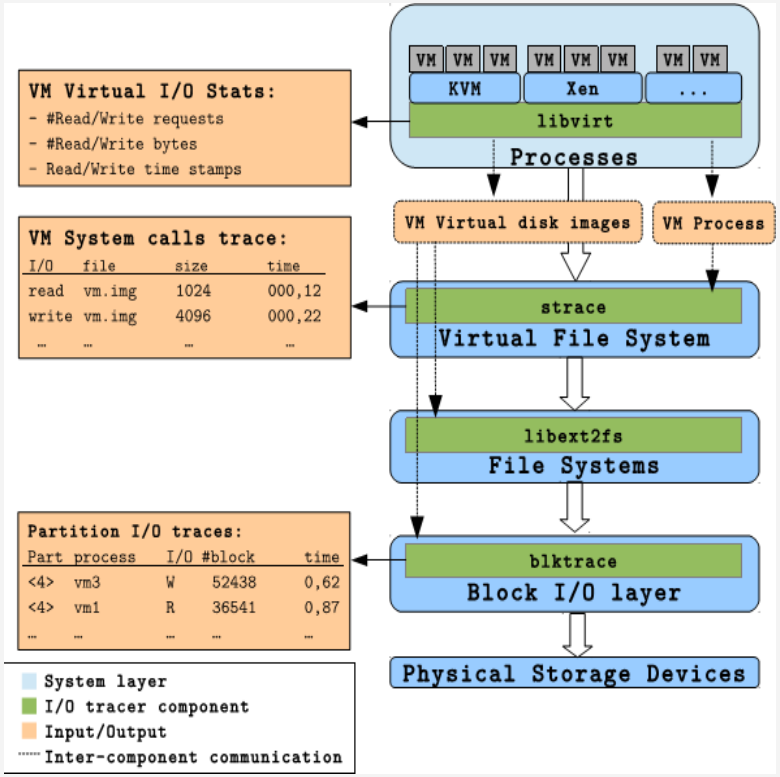


- J. Boukhobza, P.Olivier, **Flash Memory Integration: Performance and Energy Issues**, 1st Edition. ISTE Press - Elsevier 2017, ISBN 978-1-78548-124-6

1) Monitoring I/Os - first try: (re)using off the shelf tools



- A multilevel VM I/O monitoring tool
- Each level provides a different view of the I/Os
  - ***libvirt*** (https://libvirt.org/) API to get I/O statistics in hypervisor level
  - ***strace*** (see the man) to trace VM I/O system calls
  - ***libext2fs*** (http://www.giis.co.in/libext2fs.pdf) to get the mapping between VM files and corresponding blocks on the physical block device
  - ***blktrace*** (see the man) to trace I/O operations at the block level
- Parsing and filtering monitoring outputs



- H. Ouarnoughi, J. Boukhobza, F. Singhoff, and S. Rubini, "**A multi-level I/O tracer for timing and performance storage systems in IaaS Cloud, Real-time and distributed computing in emerging applications**", **IEEE REACTION**, pp.1-8, Rome, Dec. 2014.

1) Monitoring I/Os - first try: (re)using off the shelf tools -2-

- **Libvirt** → which VM uses which file
- **strace** → what system calls were launched
- **Libext2fs** → which files correspond to which blocks ?
- **Blktrace** → what blocks are accessed on the devices ?

- Traces are grouped using the **time stamp** of each trace level to have a system I/O snapshot
- Traces in different levels can be grouped according to the time to flush dirty data in Linux (*dirty_expire_centisecs* and *dirty_writeback_centisecs*)
- Issues:
  - Complexity (several tools)
  - Overhead (huge if I/O intensive apps)

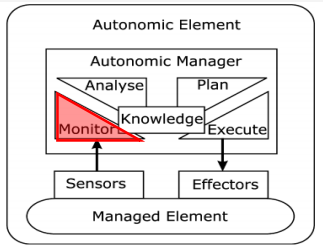
vda	4135	55024640	223	46878720	1411477037
vda	4135	55024640	272	67342336	1411477038
vda	4135	55024640	280	67932160	1411477039
vda	4135	55024640	280	67932160	1411477040

read	7	anon_inode:[signalfd]	128	1411477037
read	8	anon_inode:[eventfd]	512	1411477037
read	23	/dev/net/tun	69632	1411477038
read	23	/dev/net/tun	69632	1411477038
write	16	/var/lib/libvirt/qemu/vm1.monitor	56	1411477038
write	16	/var/lib/libvirt/qemu/vm1.monitor	56	1411477038
write	16	/var/lib/libvirt/qemu/vm1.monitor	56	1411477038
write	16	/var/lib/libvirt/qemu/vm1.monitor	56	1411477038
write	16	/var/lib/libvirt/qemu/vm1.monitor	535	1411477038
read	17	anon_inode:[eventfd]	8	1411477038
read	11	pipe	16	1411477038
write	9	anon_inode:[eventfd]	8	1411477038

<6>	22544	Q	WS	54420224	[kvm]	1411477037
<6>	22544	Q	WS	54420352	[kvm]	1411477037
<6>	22544	Q	WS	54420480	[kvm]	1411477037
<6>	22544	Q	WS	54420608	[kvm]	1411477038
<6>	22544	Q	WS	54420736	[kvm]	1411477038
<6>	22544	Q	WS	54420864	[kvm]	1411477038
<6>	22544	Q	WS	54420992	[kvm]	1411477038
<6>	22581	Q	WS	54429184	[kvm]	1411477038
<6>	22544	Q	WS	54421120	[kvm]	1411477038
<6>	22581	Q	WS	54429312	[kvm]	1411477038
<6>	22544	Q	WS	54421248	[kvm]	1411477038
<6>	22581	Q	WS	54429440	[kvm]	1411477038
<6>	22591	Q	WS	47807180	[kvm]	1411477038
<6>	22590	Q	WS	47808395	[kvm]	1411477038
<6>	22544	Q	WS	54421376	[kvm]	1411477038
<6>	22593	Q	WS	47807185	[kvm]	1411477038
<6>	22581	Q	WS	54429568	[kvm]	1411477038
<6>	22552	Q	WS	47807226	[kvm]	1411477038
<6>	22551	Q	WS	47807354	[kvm]	1411477038

- H. Ouarnoughi, J. Boukhobza, F. Singhoff, and S. Rubini, "**A multi-level I/O tracer for timing and performance storage systems in IaaS Cloud, Real-time and distributed computing in emerging applications**", **IEEE REACTION**, pp.1-8, Rome, Dec. 2014.

1) Monitoring I/Os – second try: using a full custom tool (userland+kernelland)



https://eztrace.gitlab.io/eztrace/

User-space I/O tracer

F. Trahay

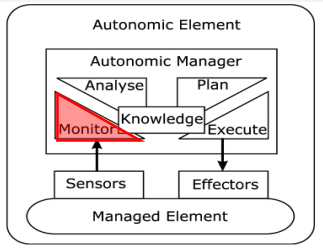
Trace Merger

Kernel-space I/O tracer

https://github.com/medislam/IOTracer

- M. I. Naas, F. Trahay, A. Colin, P. Olivier, S. Rubini, F. Singhoff, J. Boukhobza, , **EZIOTracer: Unifying Kernel and User Space I/O Tracing for Data-Intensive Applications. ACM SIGOPS Oper. Syst. Rev**. 55(1): 88-98 (2021)

1) Monitoring I/Os – second try: using a full custom tool (userland+kernelland)

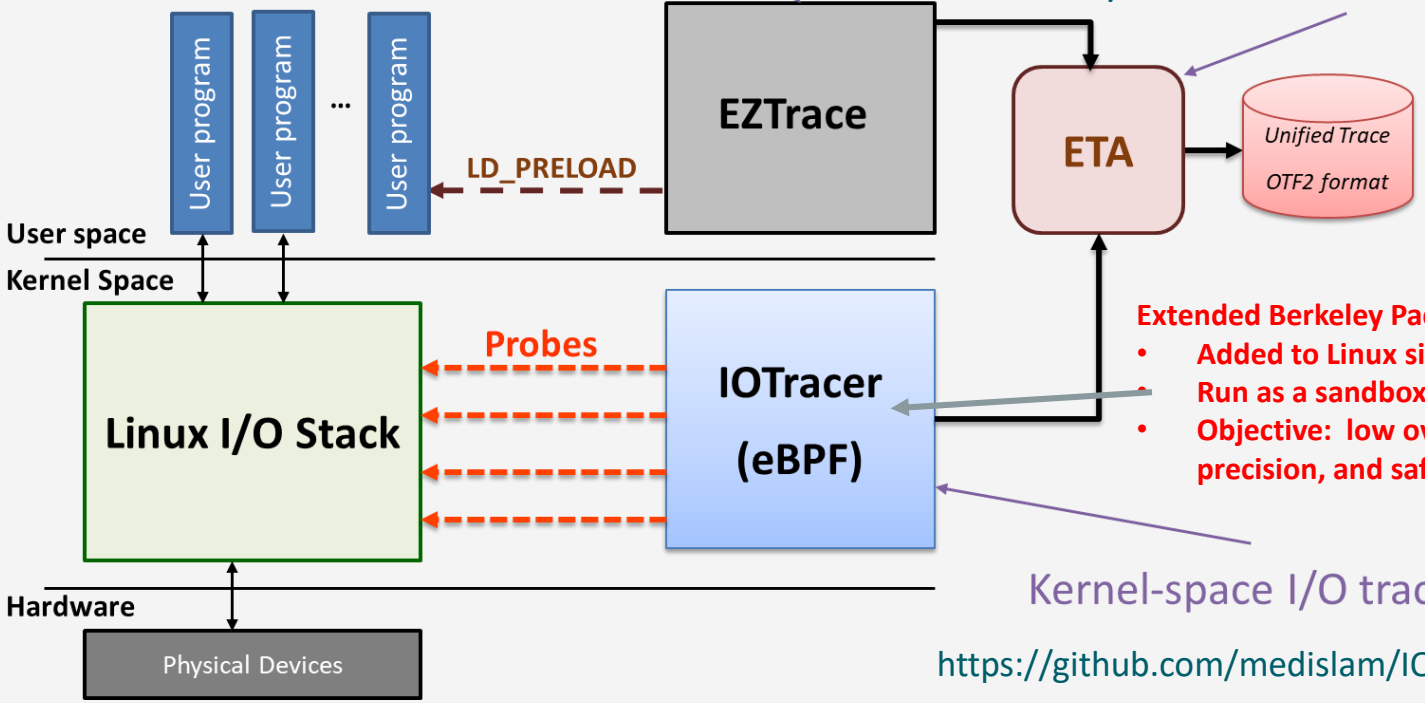


https://eztrace.gitlab.io/eztrace/

User-space I/O tracer

F. Trahay

Trace Merger



Extended Berkeley Packet Filters (eBPF)

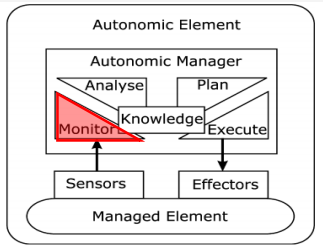
- **Added to Linux since version 3.15.**
- **Run as a sandboxed program.**
- **Objective: low overhead, high precision, and safety;**

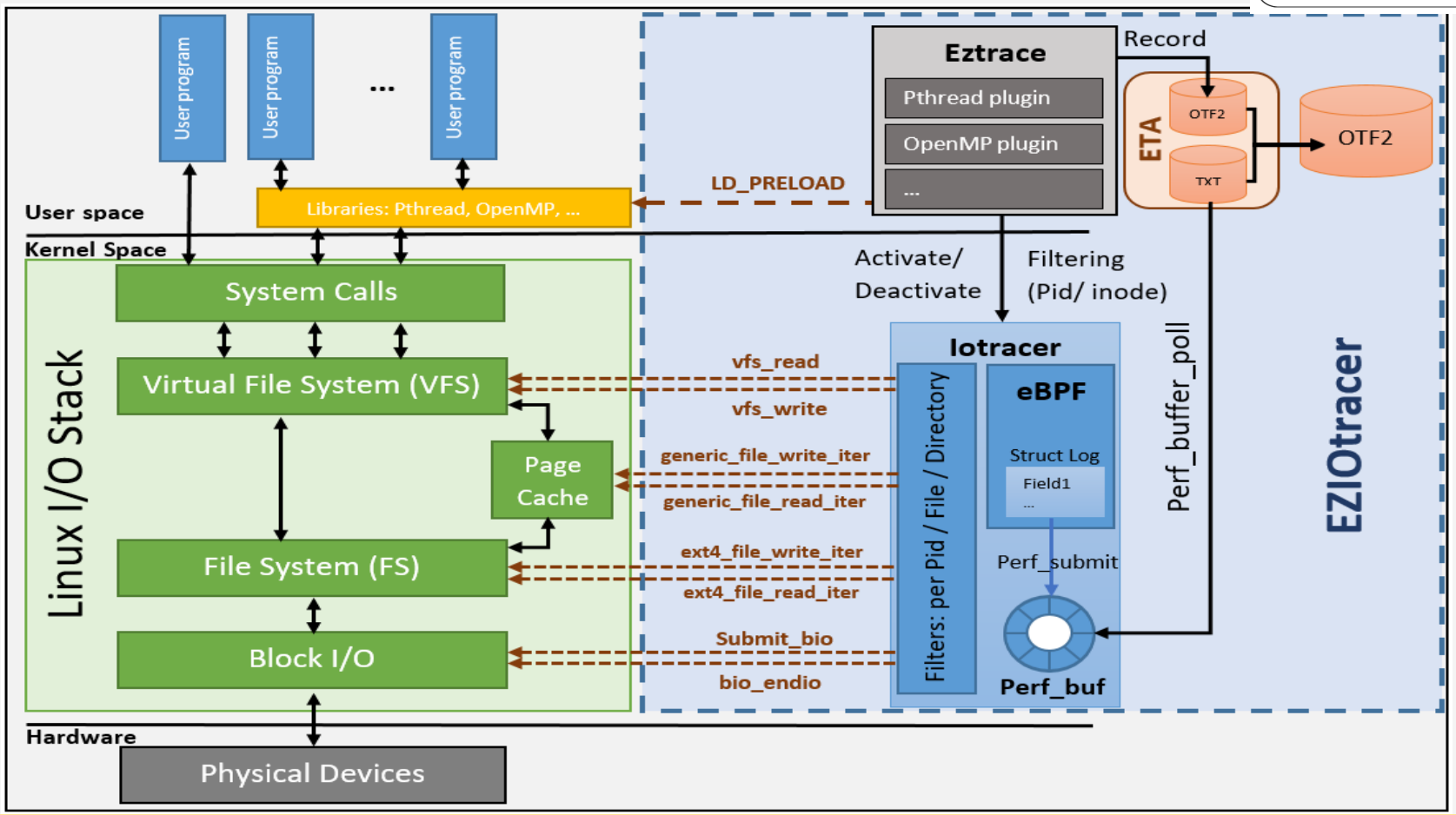
Kernel-space I/O tracer

https://github.com/medislam/IOTracer

- M. I. Naas, F. Trahay, A. Colin, P. Olivier, S. Rubini, F. Singhoff, J. Boukhobza, , **EZIOTracer: Unifying Kernel and User Space I/O Tracing for Data-Intensive Applications. ACM SIGOPS Oper. Syst. Rev**. 55(1): 88-98 (2021)

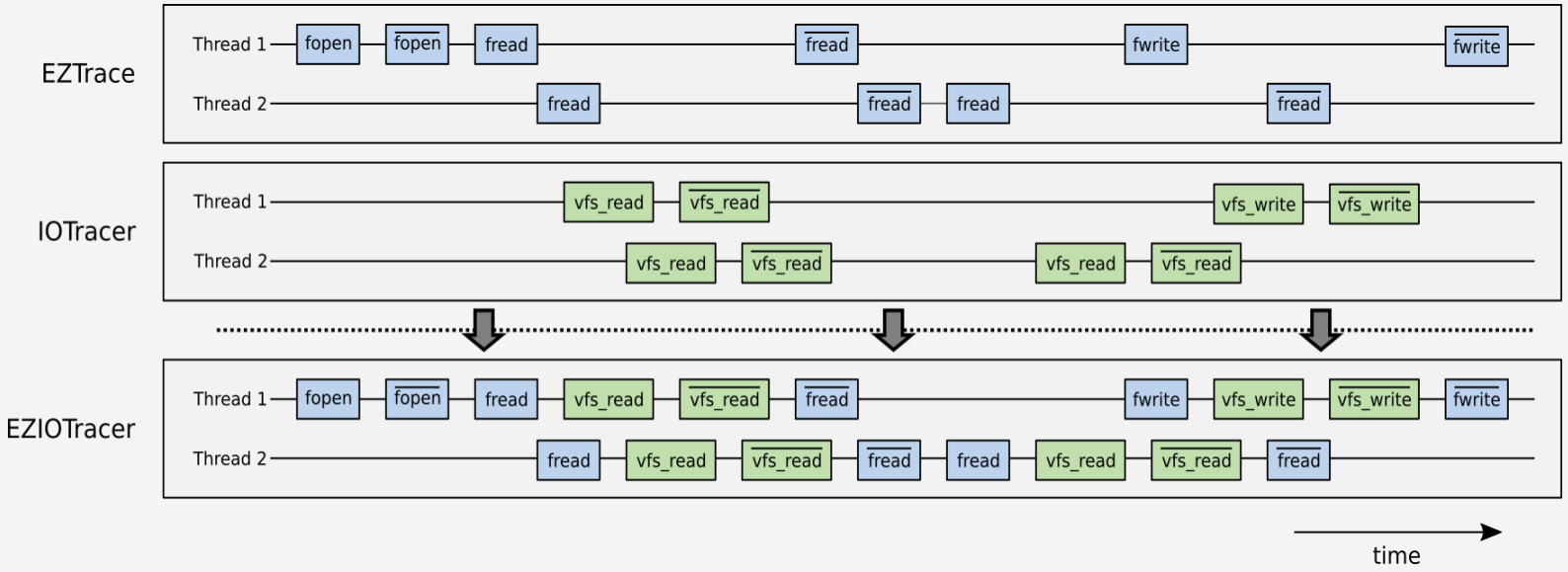
1) Monitoring I/Os – second try: using a full custom tool (userland+kernelland)





- M. I. Naas, F. Trahay, A. Colin, P. Olivier, S. Rubini, F. Singhoff, J. Boukhobza, , **EZIOTracer: Unifying Kernel and User Space I/O Tracing for Data-Intensive Applications. ACM SIGOPS Oper. Syst. Rev**. 55(1): 88-98 (2021)

1) Monitoring I/Os – second try: using a full custom tool (userland+kernelland)

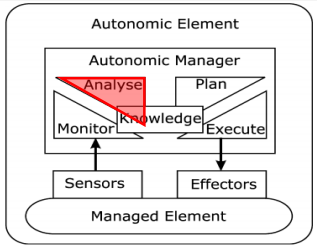


- M. I. Naas, F. Trahay, A. Colin, P. Olivier, S. Rubini, F. Singhoff, J. Boukhobza, , **EZIOTracer: Unifying Kernel and User Space I/O Tracing for Data-Intensive Applications. ACM SIGOPS Oper. Syst. Rev**. 55(1): 88-98 (2021)

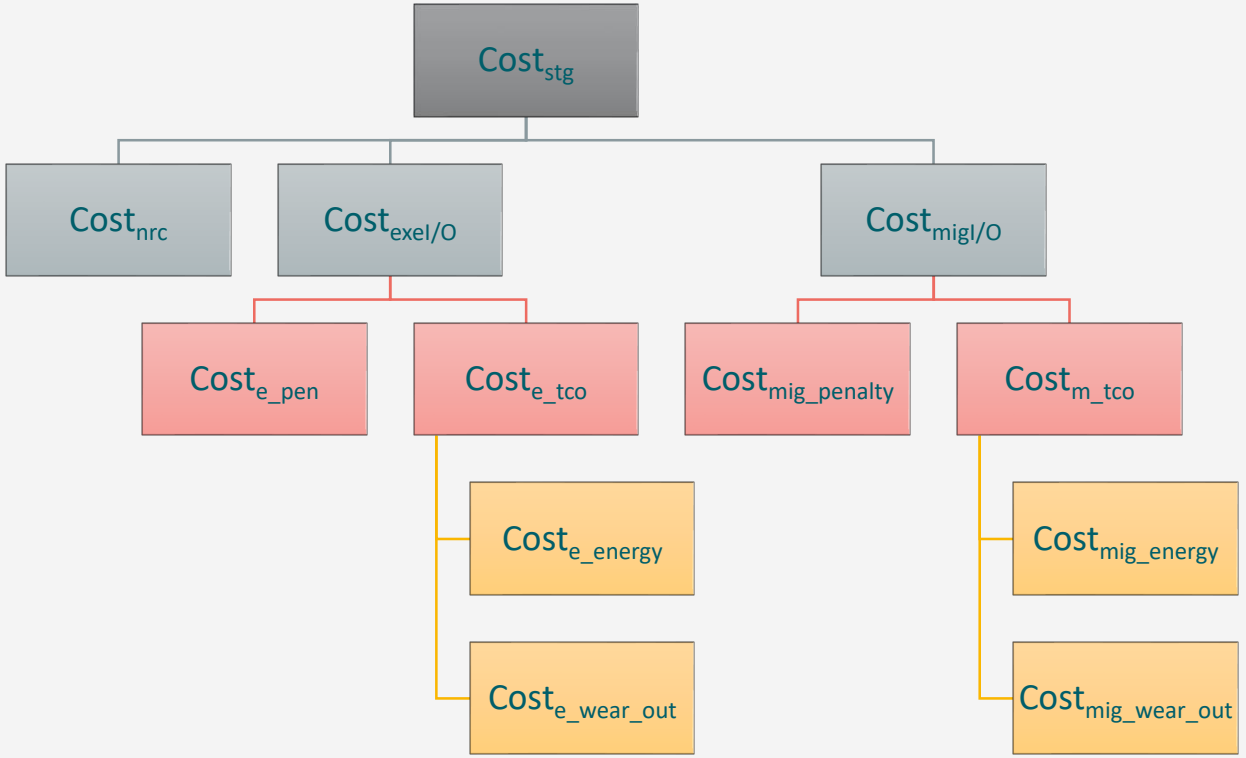
Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

The « Analyze » step



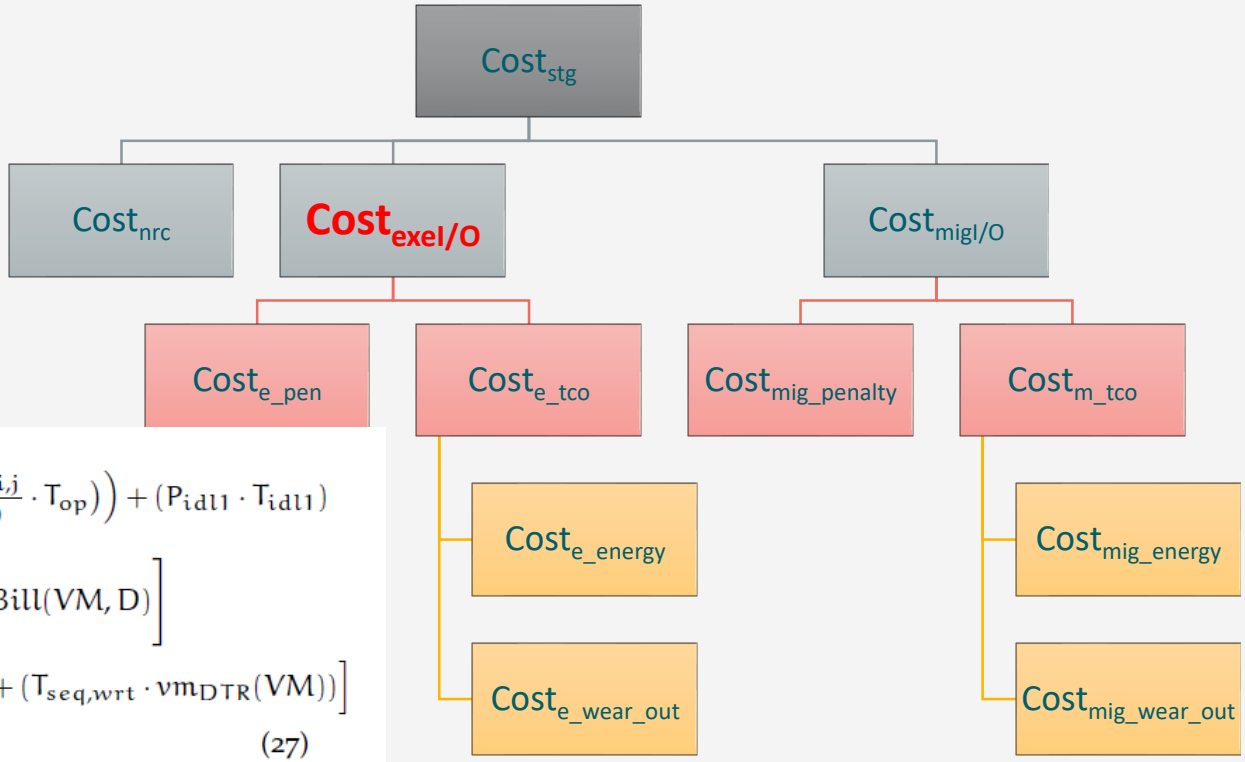
- Evaluate the storage cost for VMs
  - Execution/migration, energy/performance, wear out, I/O patterns



- H. Ouarnoughi, J. Boukhobza, F. Singhoff, S. Rubini, "**A Cost Model for Virtual Machine Storage in Cloud IaaS**", in proceedings of the EUROMICRO International conference on Parallel, Distributed, and Network based processing (**EUROMICRO PDP**), pp. 664-671, Heraklion, Feb. 2016.

The « Analyze » step for IaaS I/Os

- Evaluate the storage cost for VMs
  - Execution/migration, energy/performance, wear out, I/O patterns



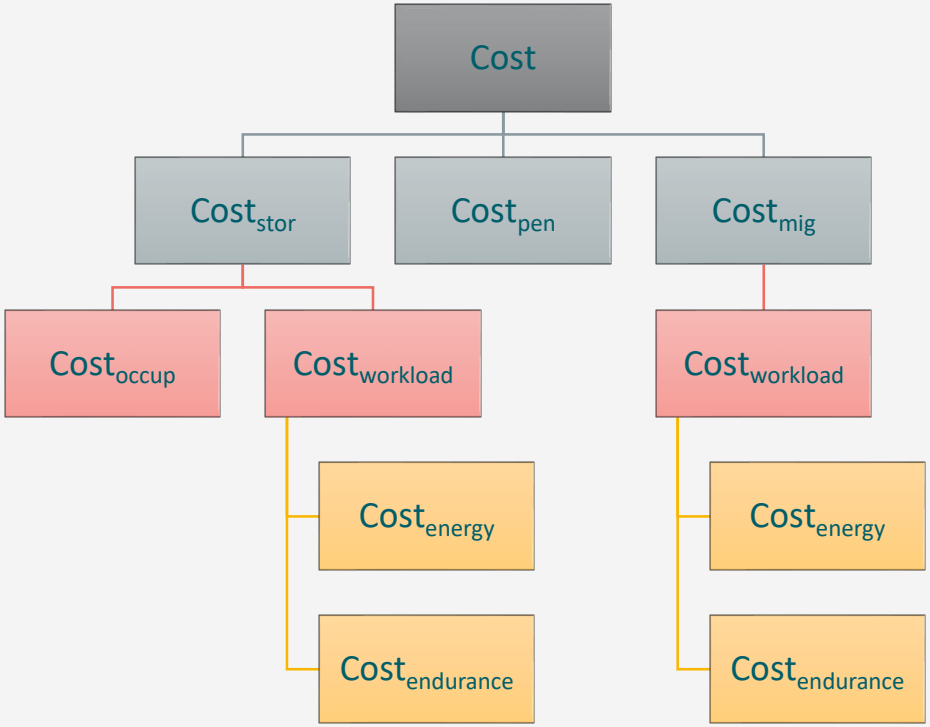
$$\mathrm{Cost}_{exe}(VM,D) = \left[\left[\sum_{i}^{seq,rnd}\ \sum_{j}^{read,wrt}\left(P_{i,j}\cdot\left(\frac{rate_{i,j}}{100}\cdot T_{op}\right)\right) + (P_{idl1}\cdot T_{idl1}) + (P_{idl2}\cdot T_{idl2})\right]\cdot E_{UP}\right] + \left[\left(1-\frac{dev_{DTR}(D)}{req_{DTR}(VM)}\right)\cdot Bill(VM,D)\right] + \left[\frac{stg_{UP}\cdot stg_{cap}}{MAX_{wrt}}\right]\cdot\left[(T_{rnd,wrt}\cdot rate_{IO}\cdot req_{size}) + (T_{seq,wrt}\cdot vm_{DTR}(VM))\right] \quad (27)$$

- H. Ouarnoughi, J. Boukhobza, F. Singhoff, S. Rubini, "**A Cost Model for Virtual Machine Storage in Cloud IaaS**", in proceedings of the EUROMICRO International conference on Parallel, Distributed, and Network based processing (**EUROMICRO PDP**), pp. 664-671, Heraklion, Feb. 2016.

The « Analyze » step for DBaaS I/Os

/ Evaluate the storage cost DBaaS objects

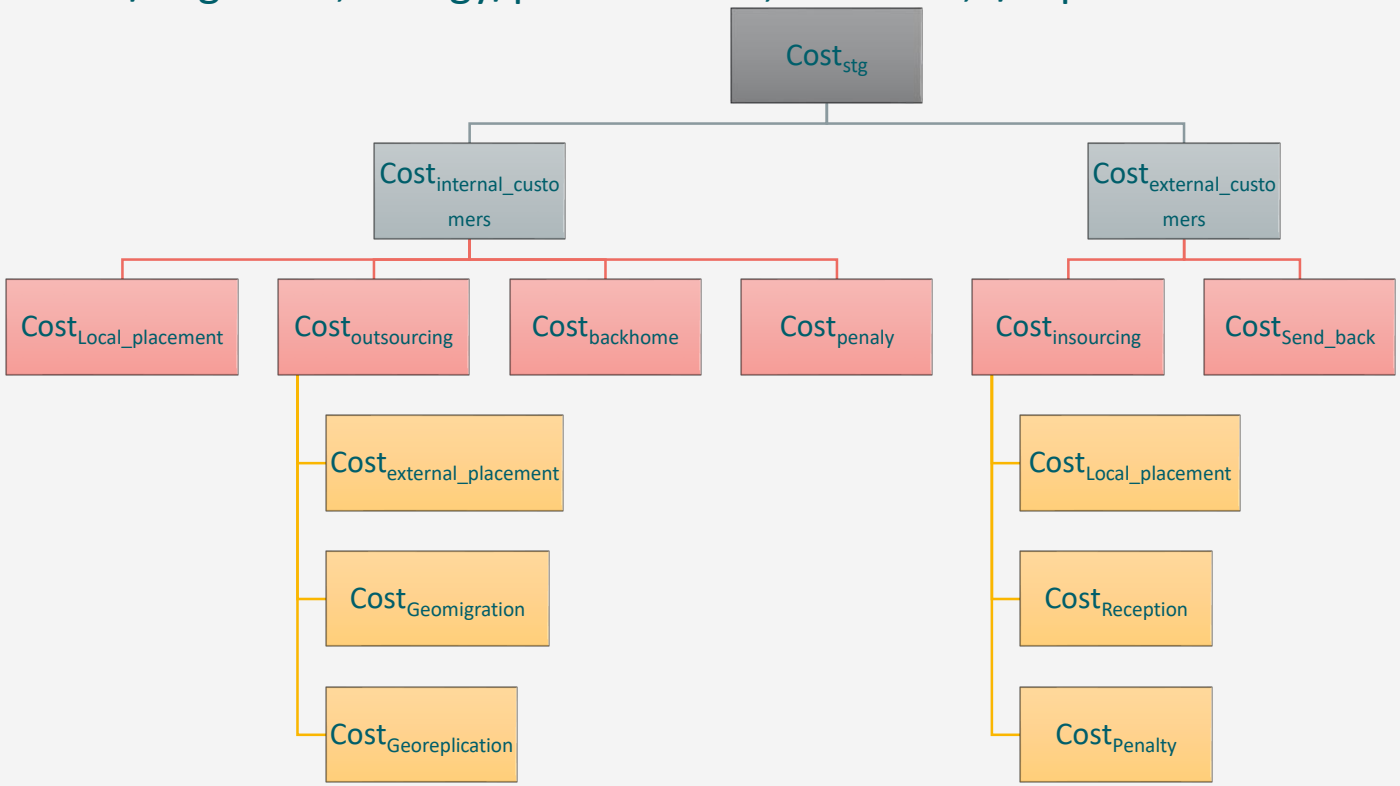
/ Storage / Penalty / Migration



- D. Boukhelef, J. Boukhobza, K. Boukhalfa, "**A Cost Model for DBaaS Storage**", 27th International Conference on Database and Expert Systems Applications (**DEXA**), pp. 223-240, Porto, Sep. 2016.

The « Analyze » step for DBaaS I/Os in Cloud Federations

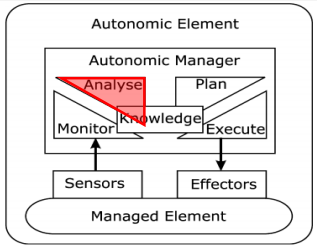
- Evaluate the storage cost DBaaS objects in a Cloud federation
  - Execution/migration, energy/performance, wear out, I/O patterns



- Amina Chikhaoui, Kamel Boukhalfa, Jalil Boukhobza, **A Cost Model for Hybrid Storage Systems in a Cloud Federations**, Federated Conference on Computer Science and Information Systems (**FedCSIS**), Poznan, 2018

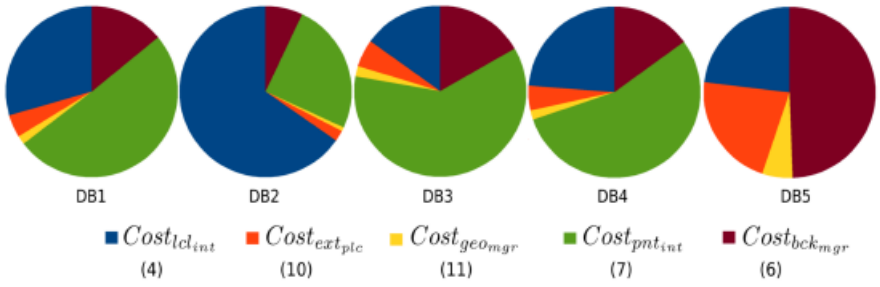
jalil.boukhobza@ensta-bretagne.fr

The « Analyze » step for DBaaS I/Os in Cloud Federations

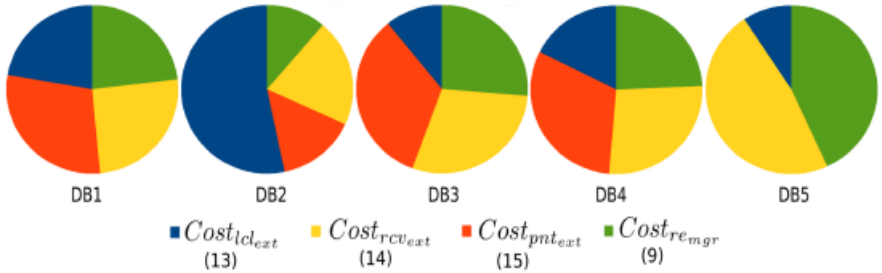


- Relevance of the modeled costs
- The importance of the **local placement (blue)** and **penalty costs (green)** of the internal customers.
- Database size → **external placement (blue), geo-migration (green)** and **back-migration**(yellow)costs.
- The storage device type and workload patterns → local placement of internal and external objects and penality costs.

- The cost model can be used for:
  - pricing strategy,
  - placement strategies,
  - Resource dimensionning



(a) Internal customers placement sub-costs



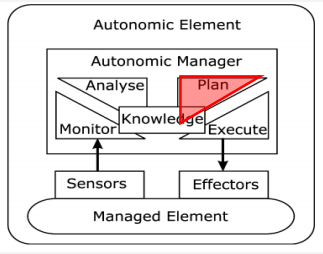
(b) External customers placement sub-costs

- Amina Chikhaoui, Kamel Boukhalfa, Jalil Boukhobza, **A Cost Model for Hybrid Storage Systems in a Cloud Federations**, Federated Conference on Computer Science and Information Systems (**FedCSIS**), Poznan, 2018

Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
  - Ephemeral resource management in the Cloud Capacity
  - Usage
  - Scheduling
- Some conclusions

The « Plan » step for centralized DBaaS



- Example: DBaaS context
  - How to place database objects in a hybrid storage system to minimize the overall cost and satisfying SLA.

$$
\begin{cases}
Minimize(Cost_{pl,T}) & \text{(a)} \\
\forall\, d_j,\ (\sum_{o_{i,u_k} \in O_{d_j}} s_{o_{i,u_k}}) \le c_{d_j} & \text{(b)} \\
\forall d_j,\ (\sum_{op \in OP} (\frac{\sum_{o_{i,u_k} \in O_{d_j}} req_{op,o_{i,u_k}}}{iops_{op,d_j}})) \le 1 & \text{(c)} \\
\forall u_k \in U,\ iops_{offered,u_k} \ge iops_{hard,u_k} & \text{(d)} \\
\forall o_{i,u_k} \in O_{u_k},\ \exists!\, d_j,\ pl(o_{i,u_k}) = d_j & \text{(e)}
\end{cases}
$$

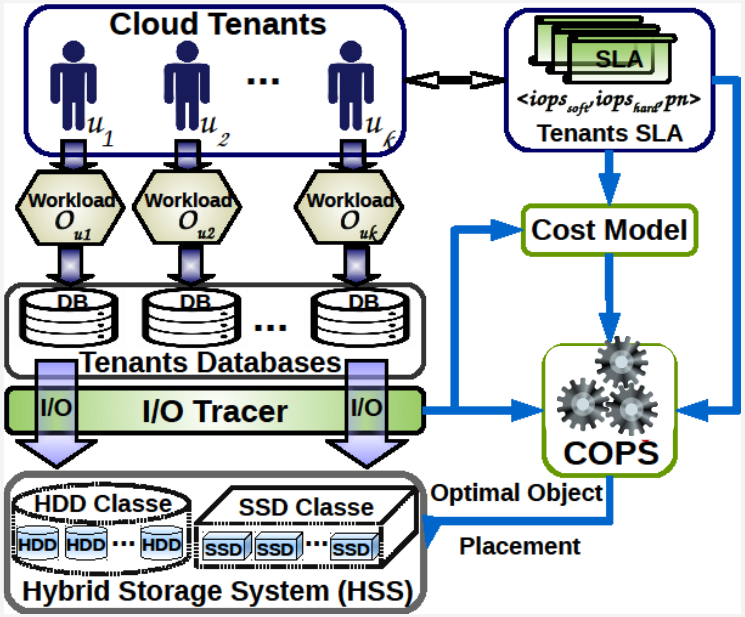
Minimize overall cost

Respecting available storage space

Respecting available storage bandwidth

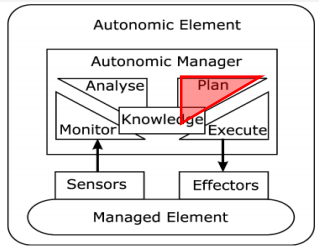
Comply with hard SLA

Store once, no replication

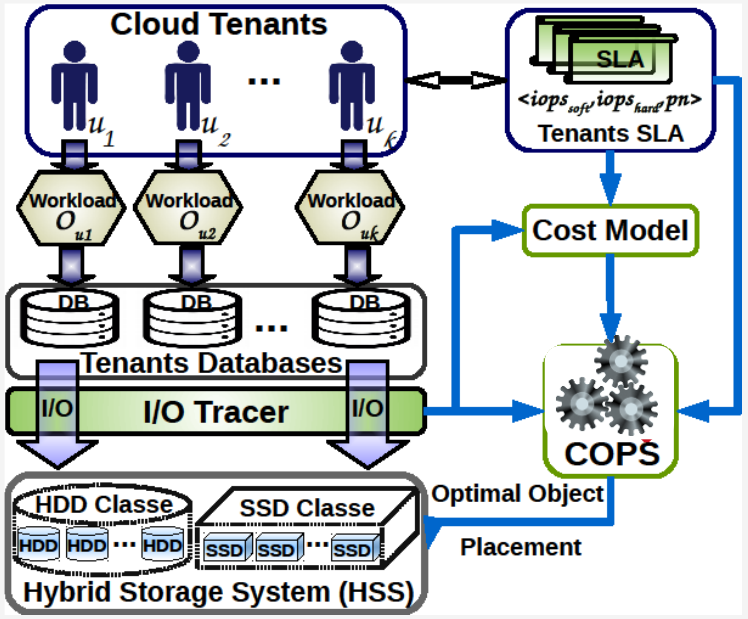


- D. Boukhelef, K. Boukhalfa, J. Boukhobza, H. Ouarnoughi, L. Lemarchand, "**COPS: Cost Based Object Placement Strategies on Hybrid Storage System for DBaaS Cloud**", The 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (**IEEE/ACM CCGRID**), May 2017.
- D. Boukhelef, J. Boukhobza, K. Boukhalfa, H. Ouarnoughi, L. Lemarchand, "**Optimizing the cost of DBaaS object placement in hybrid storage systems**", **Future Generation Computer Systems**, Elsevier, Volume 93, 2019, Pages 176-187 ,

The « Plan » step for centralized DBaaS -2-

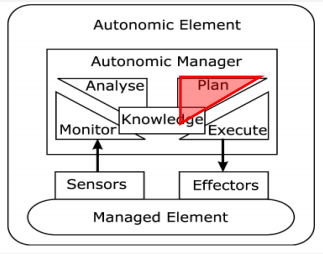


- **H-COPS**: a heuristic cost-based object placement strategy
  1. **Initialization**
     - Place objects in the cheapest storage system
  2. **Feasibility**
     - Storage constraint satisfaction
       - Move objects from overfilled devices: $\delta(o_{i,u_k}) = \frac{s_{o_{i,u_k}}}{d_{costly}(o_{i,u_k}) - d_{cheap}(o_{i,u_k})}$
     - Hard SLA constraint satisfaction
       - Move objects from HDD to SSD to **guarantee hard SLA**: $\lambda(o_{i,u_k}) = \frac{t_{exec}(o_{i,u_k}, d_{costly}) - t_{exec}(o_{i,u_k}, d_{cheap})}{d_{costly}(o_{i,u_k}) - d_{cheap}(o_{i,u_k})}$
  3. **Optimization**
     - Tradeoffs between storage cost and penalties to **guarantee soft SLA**: $\mu(u_k) = Cost_{rnl}(u_k) - Cost_{SLA}(u_k)$



- D. Boukhelef, K. Boukhalfa, J. Boukhobza, H. Ouarnoughi, L. Lemarchand, "**COPS: Cost Based Object Placement Strategies on Hybrid Storage System for DBaaS Cloud**", The 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (**IEEE/ACM CCGRID**), May 2017.
- D. Boukhelef, J. Boukhobza, K. Boukhalfa, H. Ouarnoughi, L. Lemarchand, "**Optimizing the cost of DBaaS object placement in hybrid storage systems**", **Future Generation Computer Systems**, Elsevier, Volume 93, 2019, Pages 176-187 ,

The « Plan » step for Federated DBaaS



$$\min \begin{bmatrix} store(x) \\ migrate(x) \\ latecy(x) \end{bmatrix}$$ ← Minimize 3 costs

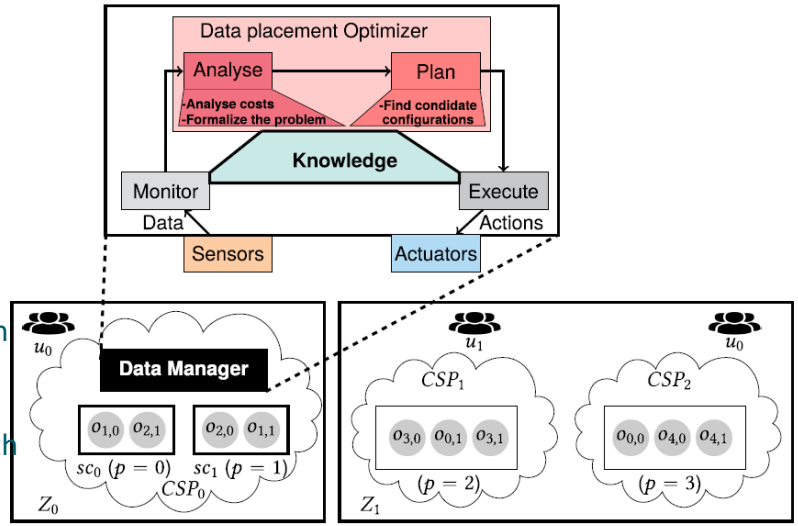
$$S.T. \quad \sum_{u_k} \sum_{o_{l,k} \in sc_j} S_{o_{l,k}} \leq csc_j \qquad \forall j < J$$ ← Respecting internal available storage space

$$\sum_{u_k} \sum_{o_{l,k} \in ss_d} S_{o_{l,k}} \leq css_d \qquad \forall d \geq J$$ ← Respecting external available storage space

$$\sum_{op \in OP} \frac{\sum_{u_k} \sum_{o_{l,k} \in sc_j} io_{o_{o_{l,k}}}(op)}{io_j(op)} \leq 1, \qquad \forall j < J$$ ← Respecting available internal storage bandwidth

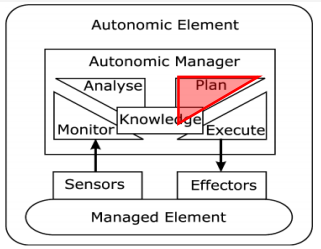
$$\sum_{op \in OP} \frac{\sum_{u_k} \sum_{o_{l,k} \in ss_d} io_{o_{o_{l,k}}}(op)}{io_d} \leq 1, \qquad \forall d \geq J$$ ← Respecting available external storage bandwidth

$$ioOffered_k >= ioHard_k \qquad \forall u_k \in U$$ ← Comply with hard SLA

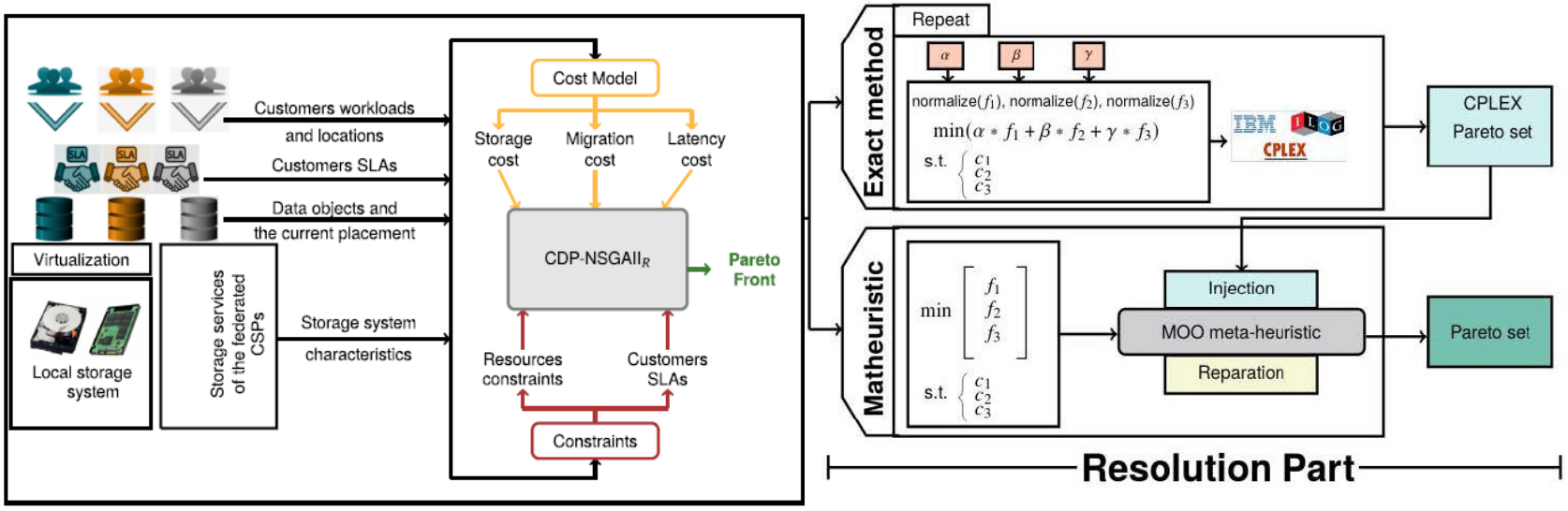


- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **Multi-objective Optimization of Data Placement in a Storage-as-a-Service Federated Cloud**. **ACM Trans. Storage** 17(3): 22:1-22:32 (2021)
- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **StorNIR, a Multi-Objective Replica Placement Strategy for Cloud Federations**, in Proceedings of the ACM SIGAP Symposium of Applied Computing (**ACM SAC**), 2021

The « Plan » step for Federated DBaaS -2-

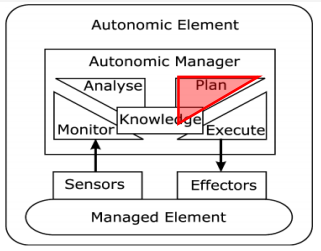


- Use metaheuristics: Non-dominated Sorting Genetic Algorithm (NSGA II)
  - Elitism
  - Polynomial complexity
- Issues for large problem instances (non feasible solutions)
- Designed **CDP-NSGAII$_{IR}$**: *Constraint Data Placement matheuristic based on NSGAII with Injection and Repair functions*
  - *Matheuristic → exact solution (Linear programming)+ NSGAII*

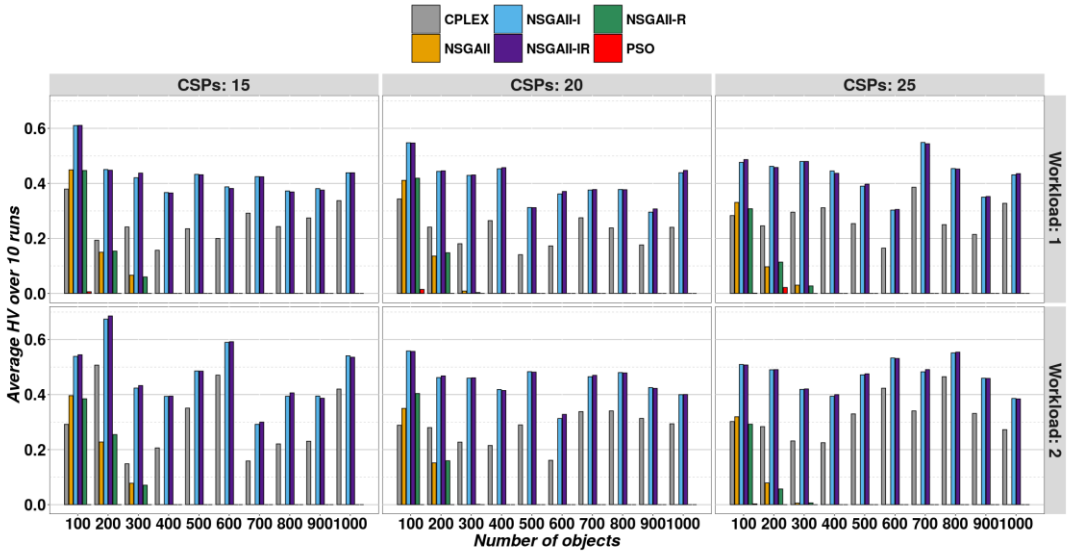


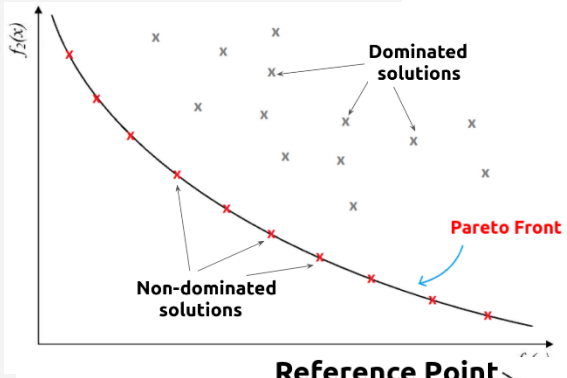
- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **Multi-objective Optimization of Data Placement in a Storage-as-a-Service Federated Cloud**. **ACM Trans. Storage** 17(3): 22:1-22:32 (2021)
- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **StorNIR, a Multi-Objective Replica Placement Strategy for Cloud Federations**, in Proceedings of the ACM SIGAP Symposium of Applied Computing (**ACM SAC**), 2021

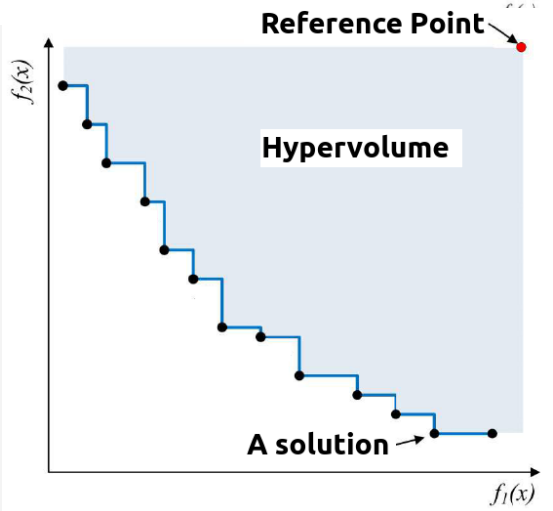
The « Plan » step for Federated DBaaS -3-



- Multiobjective solution:
  - A set of non-dominated solutions
  - Quality metrics: exhaustivity, precision, diversity
- Hypervolume:
  - Popular metric for evaluating multiobjective algorithms performance
  - the objective space (volume) dominated by all the solutions bounded by a reference point r
  - The greater the HV, the better the approximation

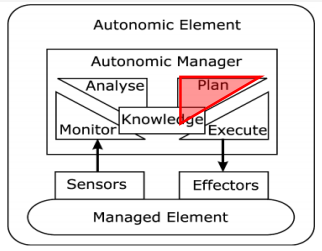




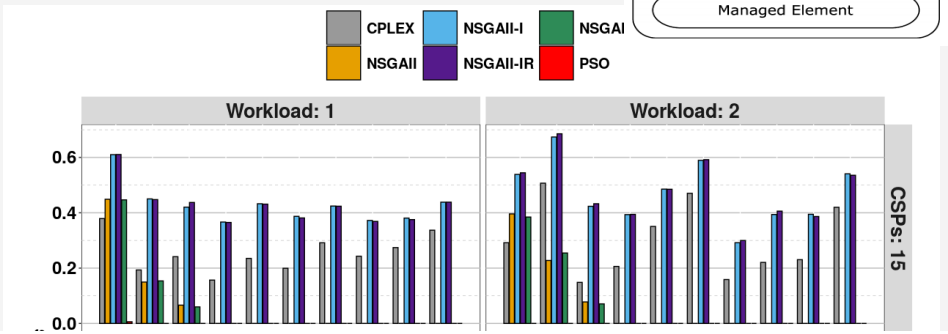


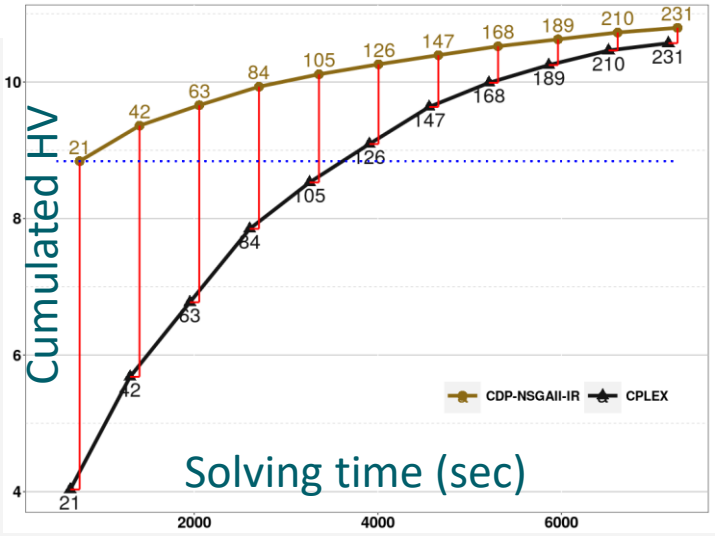
- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **Multi-objective Optimization of Data Placement in a Storage-as-a-Service Federated Cloud**. **ACM Trans. Storage** 17(3): 22:1-22:32 (2021)
- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **StorNIR, a Multi-Objective Replica Placement Strategy for Cloud Federations**, in Proceedings of the ACM SIGAP Symposium of Applied Computing (**ACM SAC**), 2021

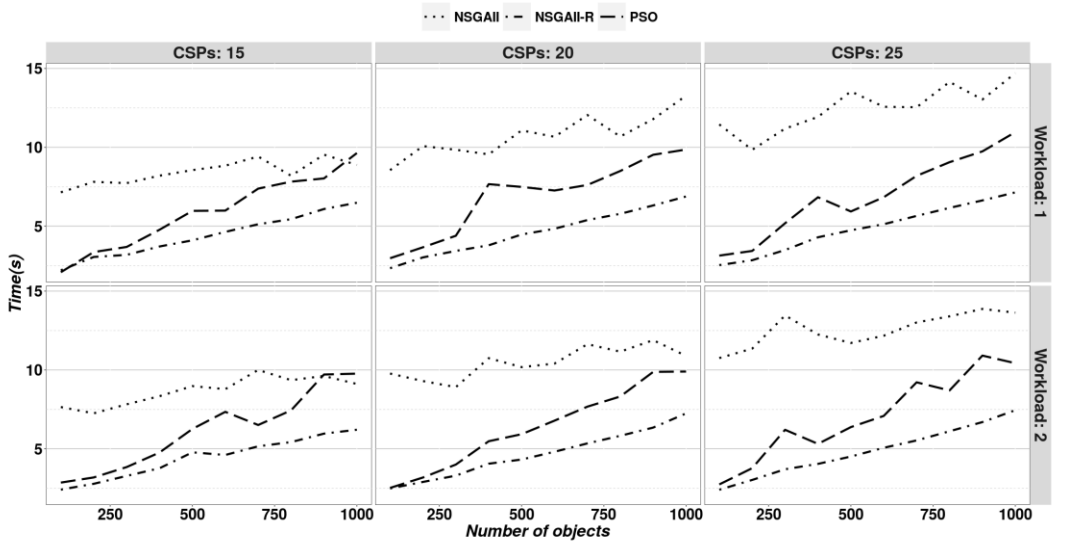
The « Plan » step for Federated DBaaS -4-



- Comparison to the exact method is not that … exact (doesn't converge → we limited the time)
- NSGAII-IR better than the others (but same as NSGAII-I … is repairing the solutions useful ? → see execution times → 40-86% less)
- Enhance injected solutions with little overhead

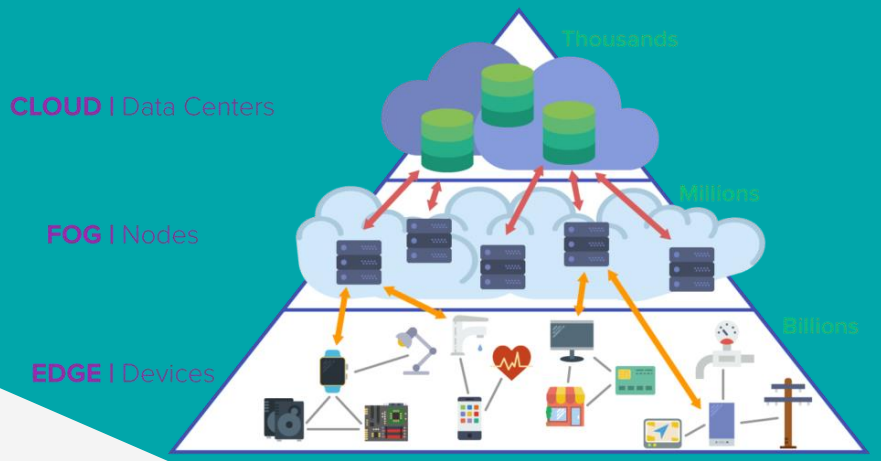






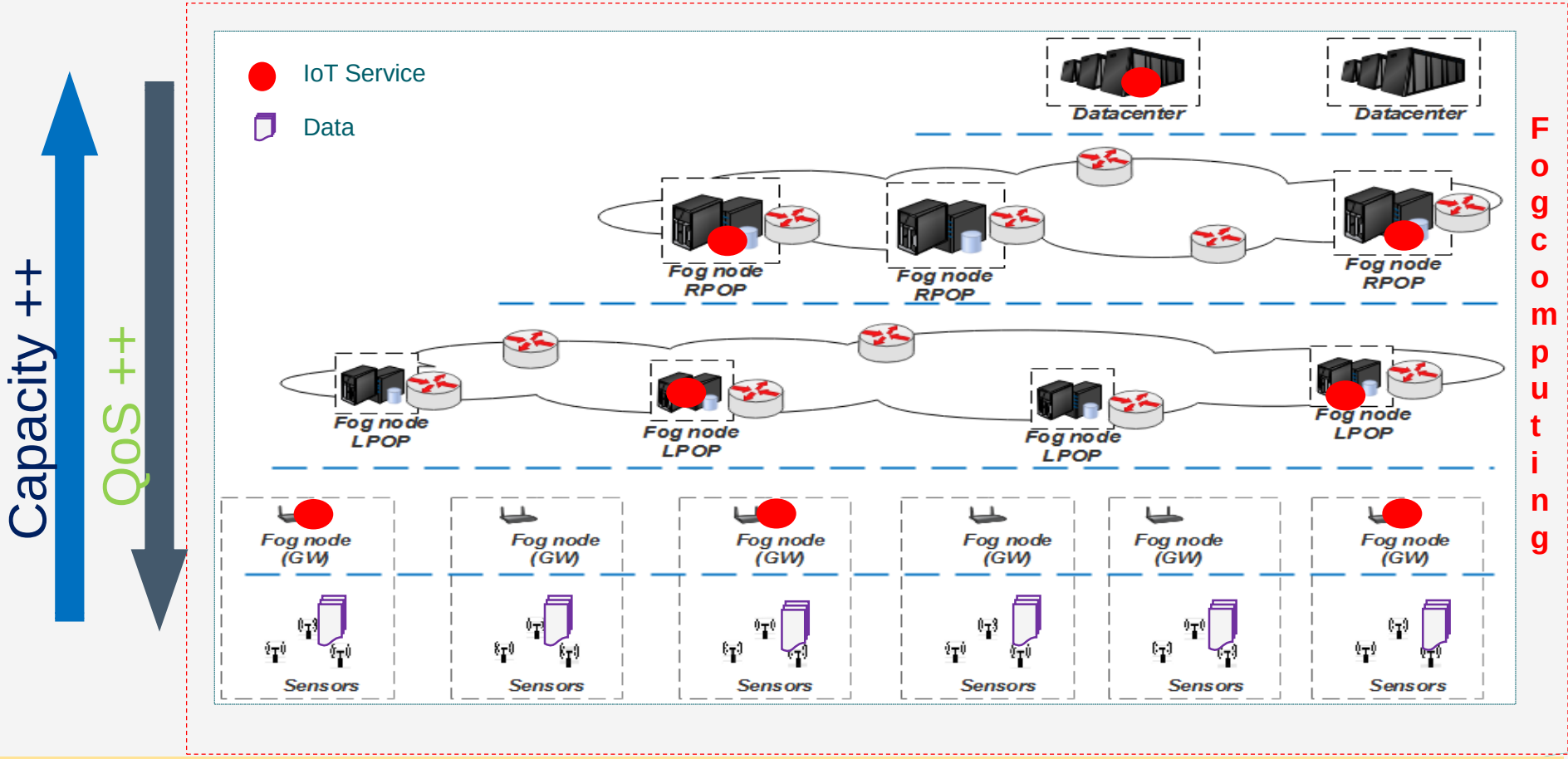
- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **Multi-objective Optimization of Data Placement in a Storage-as-a-Service Federated Cloud**. **ACM Trans. Storage** 17(3): 22:1-22:32 (2021)
- Amina Chikhaoui, Laurent Lemarchand, Kamel Boukhalfa, Jalil Boukhobza, **StorNIR, a Multi-Objective Replica Placement Strategy for Cloud Federations**, in Proceedings of the ACM SIGAP Symposium of Applied Computing (**ACM SAC**), 2021

What about Edge and Fog ?



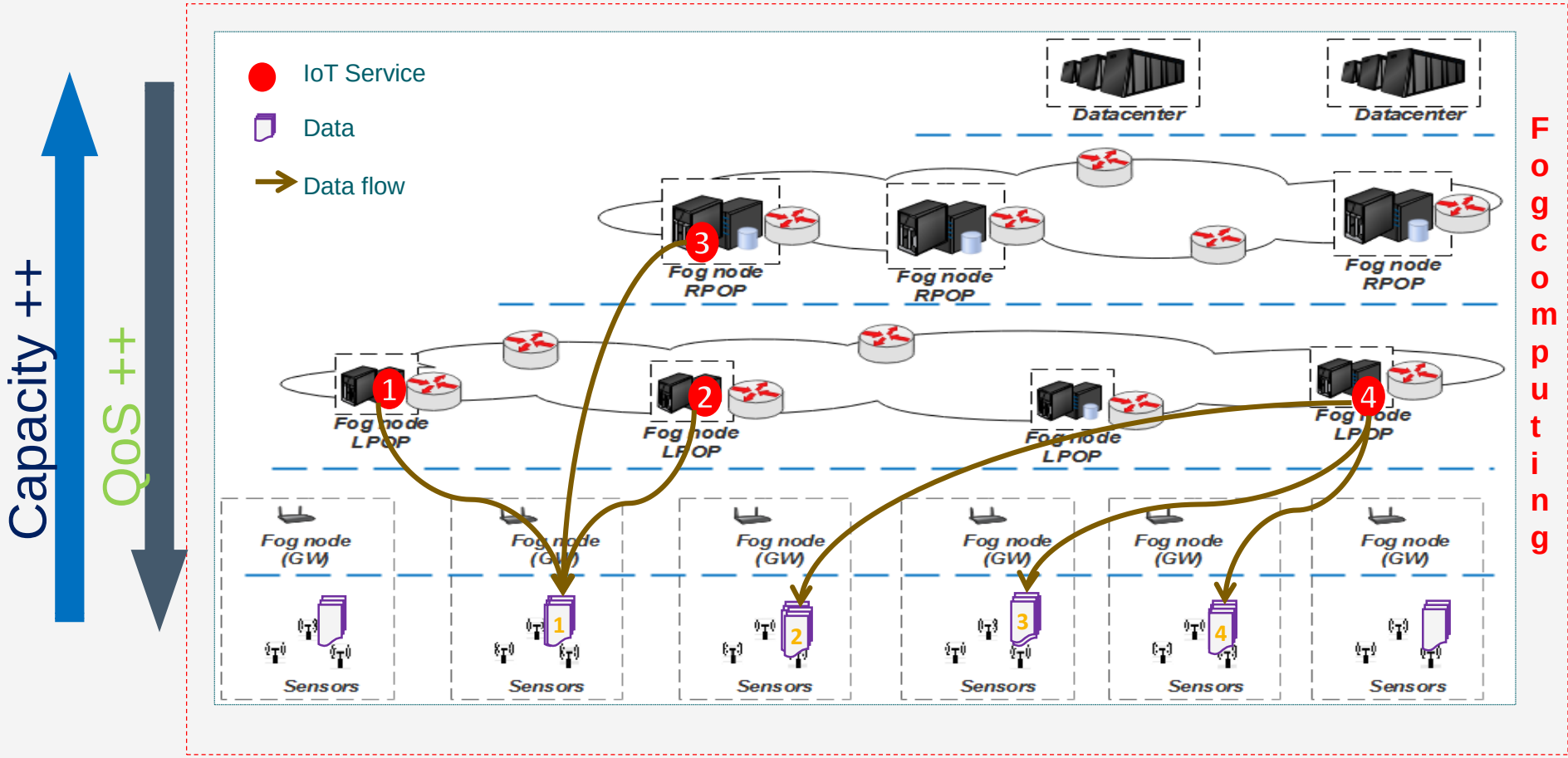
Source : https://erpinnews.com/fog-computing-vs-edge-computing

Data placement in Fog infrastructures



- Islam Naas, Philippe Raipin, Jalil Boukhobza, Laurent Lemarchand, "**iFogStor: an IoT Data Placement Strategy for Fog Infrastructure**", the IEEE International Conference on Fog and Edge Computing (**IEEE ICFEC**), pp. 97-104, Madrid, May 2017.

Data placement in Fog infrastructures



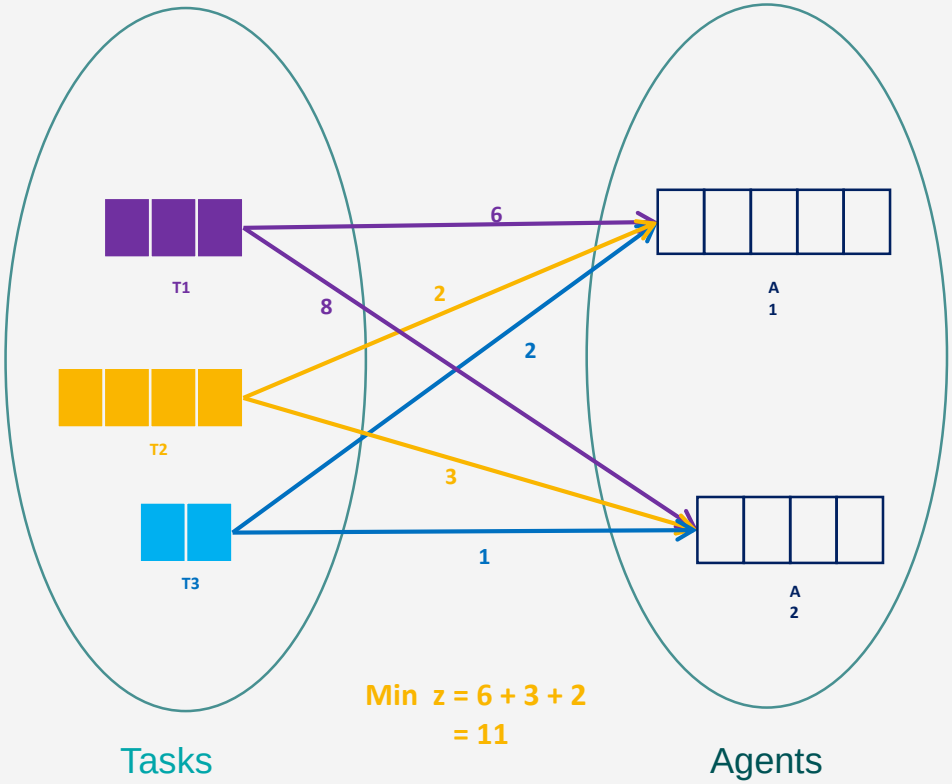
- Islam Naas, Philippe Raipin, Jalil Boukhobza, Laurent Lemarchand, **"iFogStor: an IoT Data Placement Strategy for Fog Infrastructure"**, the IEEE International Conference on Fog and Edge Computing (**IEEE ICFEC**), pp. 97-104, Madrid, May 2017.

Preliminaries: Generalized Assignment Problem (GAP)

- GAP is a NP-Hard problem

Objective: find the best assignment of n tasks to m agents while minimizing (maximizing) the overall cost Z:

1) Each task T_i has a different size $S_{i,j}$ (workload) depending to the agent A_j
2) Each task T_i has a different cost $V_{i,j}$ depending to the agent A_j
3) Each agent A_j has a capacity C_j



- Islam Naas, Philippe Raipin, Jalil Boukhobza, Laurent Lemarchand, **"iFogStor: an IoT Data Placement Strategy for Fog Infrastructure**", the IEEE International Conference on Fog and Edge Computing (**IEEE ICFEC**), pp. 97-104, Madrid, May 2017.

Approach: Data Placement Problem Modeling

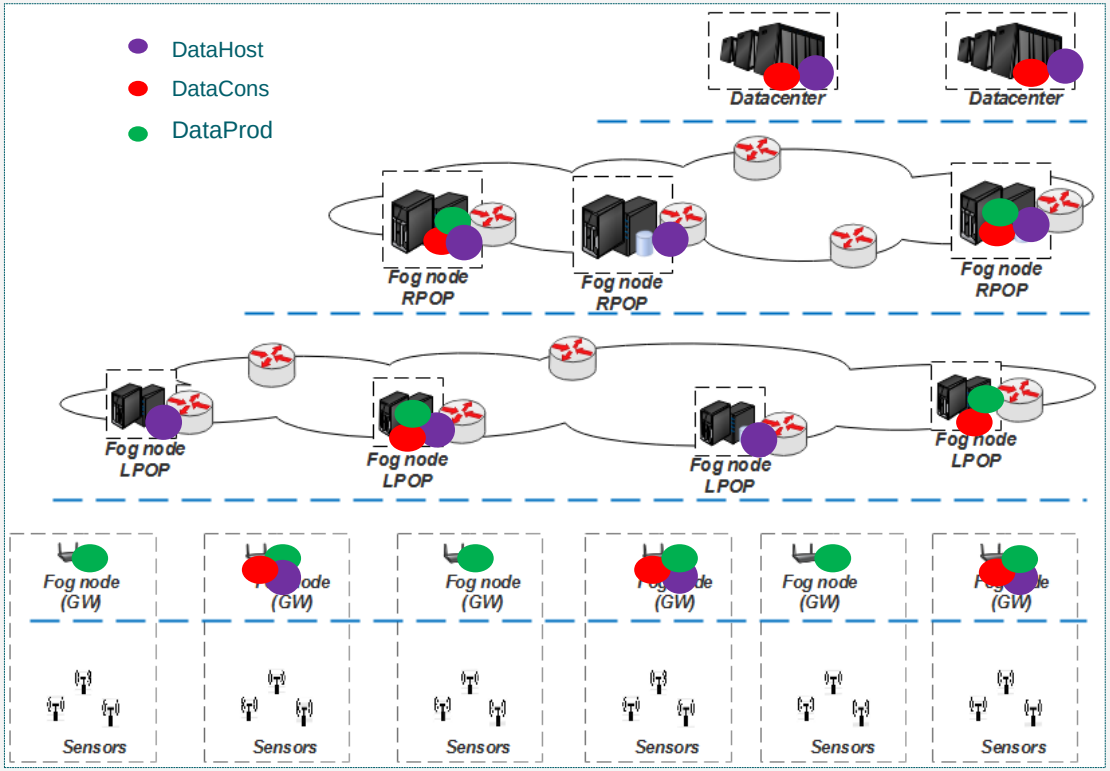
3 data actors:

- DataHost: storage equipment (with limited capacities)
- DataCons: IoT services consumers
- DataProd: IoT services producers

A network latency exists between Fog nodes

Objective: find the best location (to store data) which minimizes the overall service latency.
This turns out to finding the matrix A

$$A = \begin{array}{c} \\ d_1 \\ \vdots \\ d_l \end{array} \begin{array}{c} \begin{array}{ccc} dh_1 & \cdots & dh_n \end{array} \\ \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{l,1} & \cdots & a_{l,n} \end{bmatrix} \end{array}, \qquad a_{i,j} \in \{1,0\}$$



- Islam Naas, Philippe Raipin, Jalil Boukhobza, Laurent Lemarchand, "**iFogStor: an IoT Data Placement Strategy for Fog Infrastructure**", the IEEE International Conference on Fog and Edge Computing (**IEEE ICFEC**), pp. 97-104, Madrid, May 2017.

Approach: Data Placement Problem Modeling

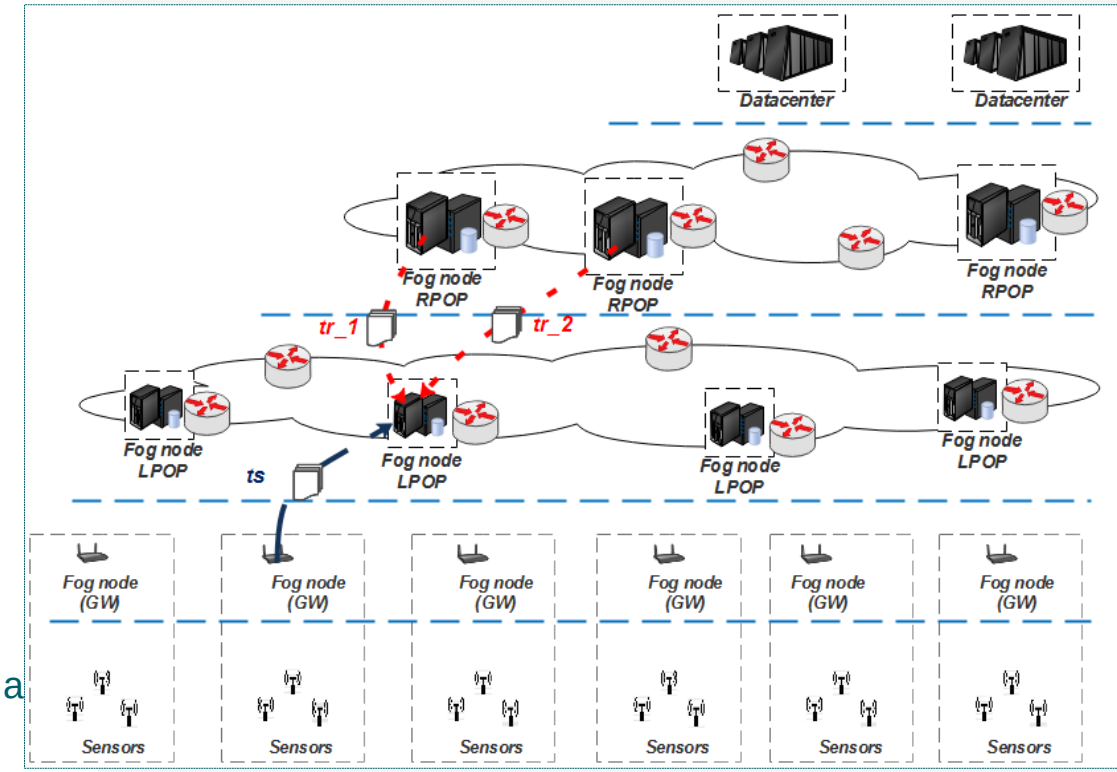
Minimize overall_latency:

$$Overall\_latency = \sum storage\ latency\ (ts) + \sum retreiving\ latency(tr)$$

(1) Storage latency (ts) has a different value according to the selected Fog node

(2) Retrieving latency (tr) has a different value according to the selected Fog node

From (1) and (2): the data placement problem is a GAP-like problem.



- Islam Naas, Philippe Raipin, Jalil Boukhobza, Laurent Lemarchand, "**iFogStor: an IoT Data Placement Strategy for Fog Infrastructure**", the IEEE International Conference on Fog and Edge Computing (**IEEE ICFEC**), pp. 97-104, Madrid, May 2017.

Approach: Data Placement Problem Modeling

Constraints:

1. The data amount assigned to a given dh_j must not exceed its free storage capacity f_{dh_j}:

$$\sum_{i\in[1..l]} s_{d_i} . a_{i,j} \leq f_{dh_j} , \forall f_{dh_j} \in FC$$

2. Each data item should be stored in one location:

$$\sum_{j\in[1..n]} a_{i,j} = 1 , \forall i \in [1..l]$$

Objective:

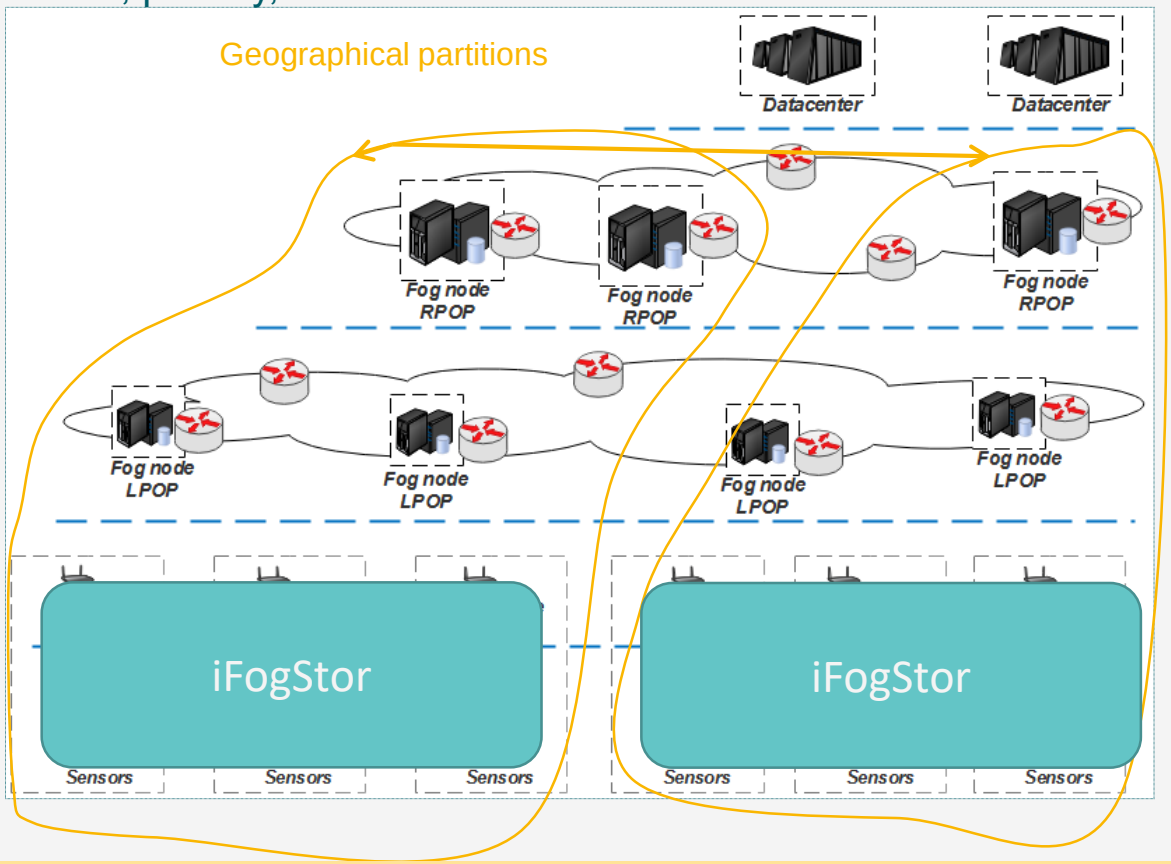
$$Minimize\ Z = \sum_{i\in[1..l]} \sum_{j\in[1..n]} \alpha_{i,j} \left(a_{i,j}\right)$$

Solved with CPLEX MILP

- Islam Naas, Philippe Raipin, Jalil Boukhobza, Laurent Lemarchand, "**iFogStor: an IoT Data Placement Strategy for Fog Infrastructure",** the IEEE International Conference on Fog and Edge Computing (**IEEE ICFEC**), pp. 97-104, Madrid, May 2017.

iFogStorZ – for Zoned Data placement in Fog infrastructures

iFogStorZ: an approximation heuristic to reduce the problem solving time based on divide and conquer → per RPOP, per city, …

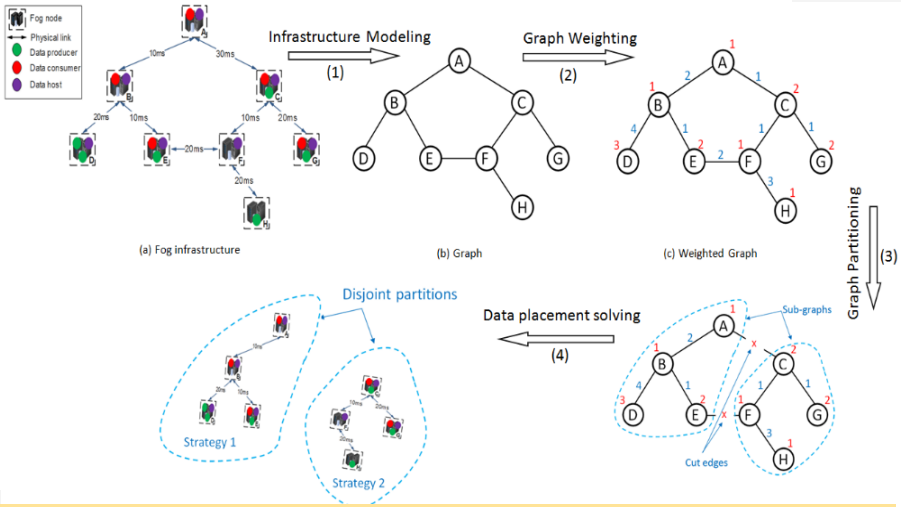


- Islam Naas, Philippe Raipin, Jalil Boukhobza, Laurent Lemarchand, "**iFogStor: an IoT Data Placement Strategy for Fog Infrastructure**", the IEEE International Conference on Fog and Edge Computing (**IEEE ICFEC**), pp. 97-104, Madrid, May 2017.

iFogStorG – for Graph-based Data placement in Fog infrastructures

- **iFogStorG**: a heuristic for data placement strategies for service latency minimization
- Subdivides the Fog infrastructure based on graph partition methods
- Subdivision process ensures:
  - **Balanced parts:** to have equivalent sub-problems (data amount & Fog nodes)
  - **Disconnected parts:** to avoid having nodes exchanging data across separate parts
- **iFogStorG** steps:
  1. Infrastructure modeling as a undirected graph
  2. Graph weighting for both vertex (nb of producers) and Edge (nb of dataflows / shortest paths)
  3. Graph partitioning (K-way partition with Metis[1]) → balance vertex weight + minimize cut edge weight
  4. Data placement solving → use iFogStor for each partition

[1] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

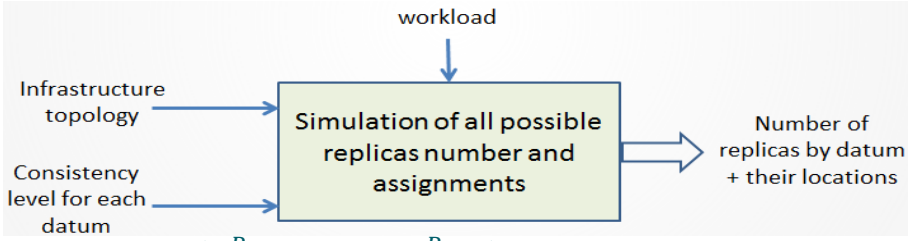


- Islam Naas, Laurent Lemarchand, Jalil Boukhobza, Philippe Raipin, "**A Graph Partitioning-based Heuristic for Runtime IoT Data Placement Strategies in a Fog infrastructure**", in The 33rd ACM/SIGAPP Symposium On Applied Computing (**ACM SAC),** Pau, Apr. 2018

iFogStorP – for P-Median based Data preplication in Fog infrastructures

- **Assumption**: If we replicate data on different parititions, we might reduce the overall latency
- Yes, but you need to deal with data consistency … 😈😈😈

Exact solution: enumerate all replica placement possibilities.



Complexity: the number of possible assignments= $d \times \left(C_n^{P_{min}} + \cdots + C_n^{P_{max}}\right)$

- d: number of data elements
- n: number of Fog nodes
- P_{min}: min number of replicas
- P_{max}: max number of replicas
- C_n^P: a combination of P from $n, C_n^P = \frac{n!}{P!(n-p)!}$

Combinatorial Explosion

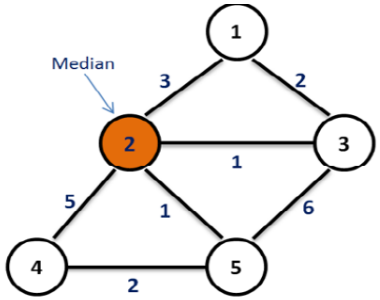
- E.g. for $n = 20, P_{min} = 3, P_{max} = 5, d = 20$ ➔ 322,335 possible assignments

- Mohammed Islam Naas, Laurent Lemarchand, Philippe Raipin, Jalil Boukhobza. **IoT Data Replication and Consistency Management in Fog computing. Journal of Grid Computing,** Springer Verlag, 2021, 19 (3), pp.33

iFogStorP - Data placement in Fog infrastructures

- **Idea**: Reduce the number of possible assignements
  - Store data only in **shortest paths** (between producers and consumers)
  - From the shortest path nodes: choose only **P-Median**

- Median: vertex for which the sum of the shortest paths costs to all other vertices is the smallest
- P-Median: generalization of the Median problem
  - Find a subset of P medians that minimizes the sum of the shortest path costs existing between medians and all others vertices.



Source	Destination	Path	Cost
1	2	1-2	3
2	2	2-2	0
3	2	3-2	1
4	2	4-5-2	3
5	2	5-2	1

For All Data :

Get All shortest paths nodes

For $P \in [P\_(\min\ ), P\_\max\ ]$, number of replicas:

Find P-median to place P replicas (using CPLEX)

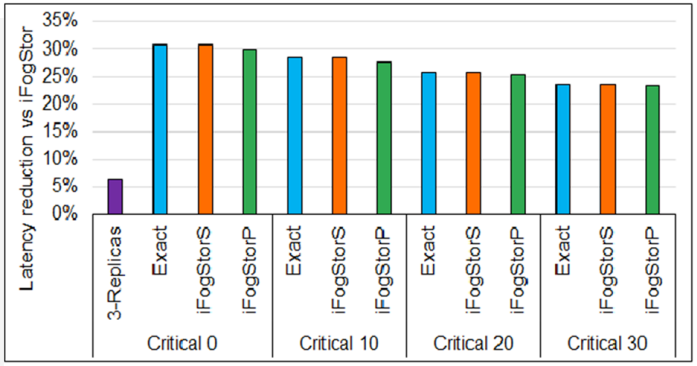
Estimate the latency overhead of this assignment (a micro simulation is done using iFogSim)

Choose P with the minimum latency overhead.

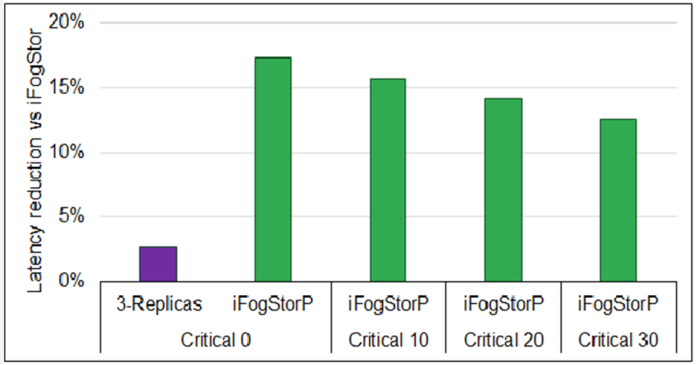
- Mohammed Islam Naas, Laurent Lemarchand, Philippe Raipin, Jalil Boukhobza. **IoT Data Replication and Consistency Management in Fog computing**. **Journal of Grid Computing**, Springer Verlag, 2021, 19 (3), pp.33

iFogStorP - Data placement in Fog infrastructures

- Compared with no replica and other solutions: exact and iFogStorS (no P-Median, only shortest paths)

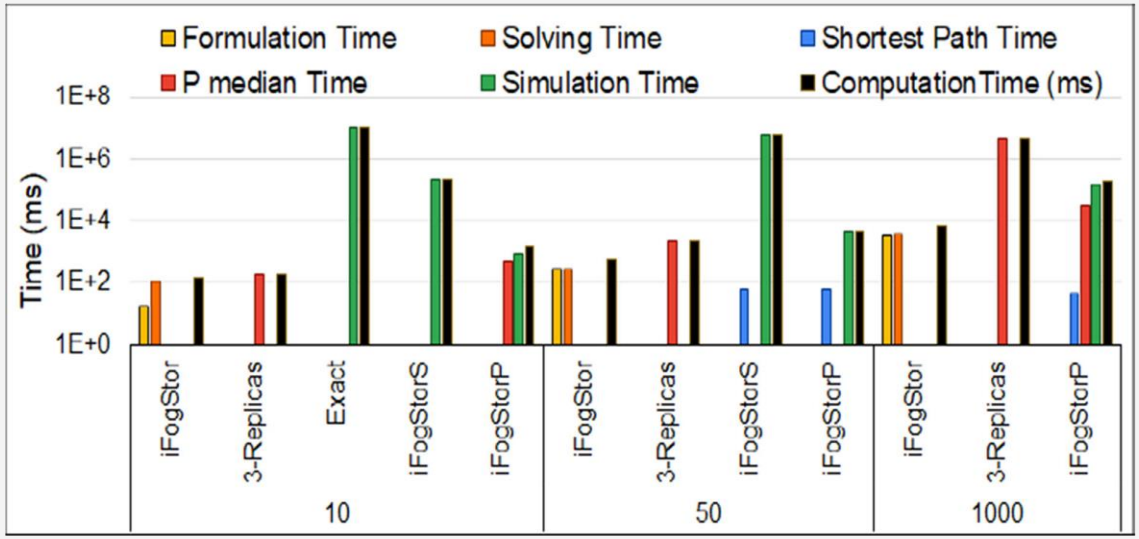


(a) Latency-reduction-10.



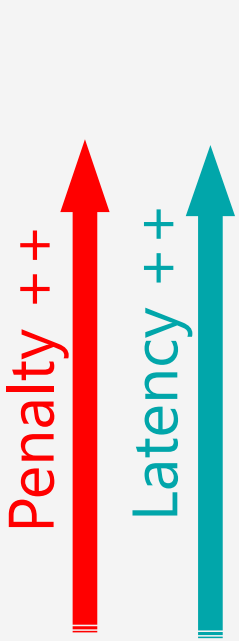
(c) Latency-reduction-1000-Zoned.

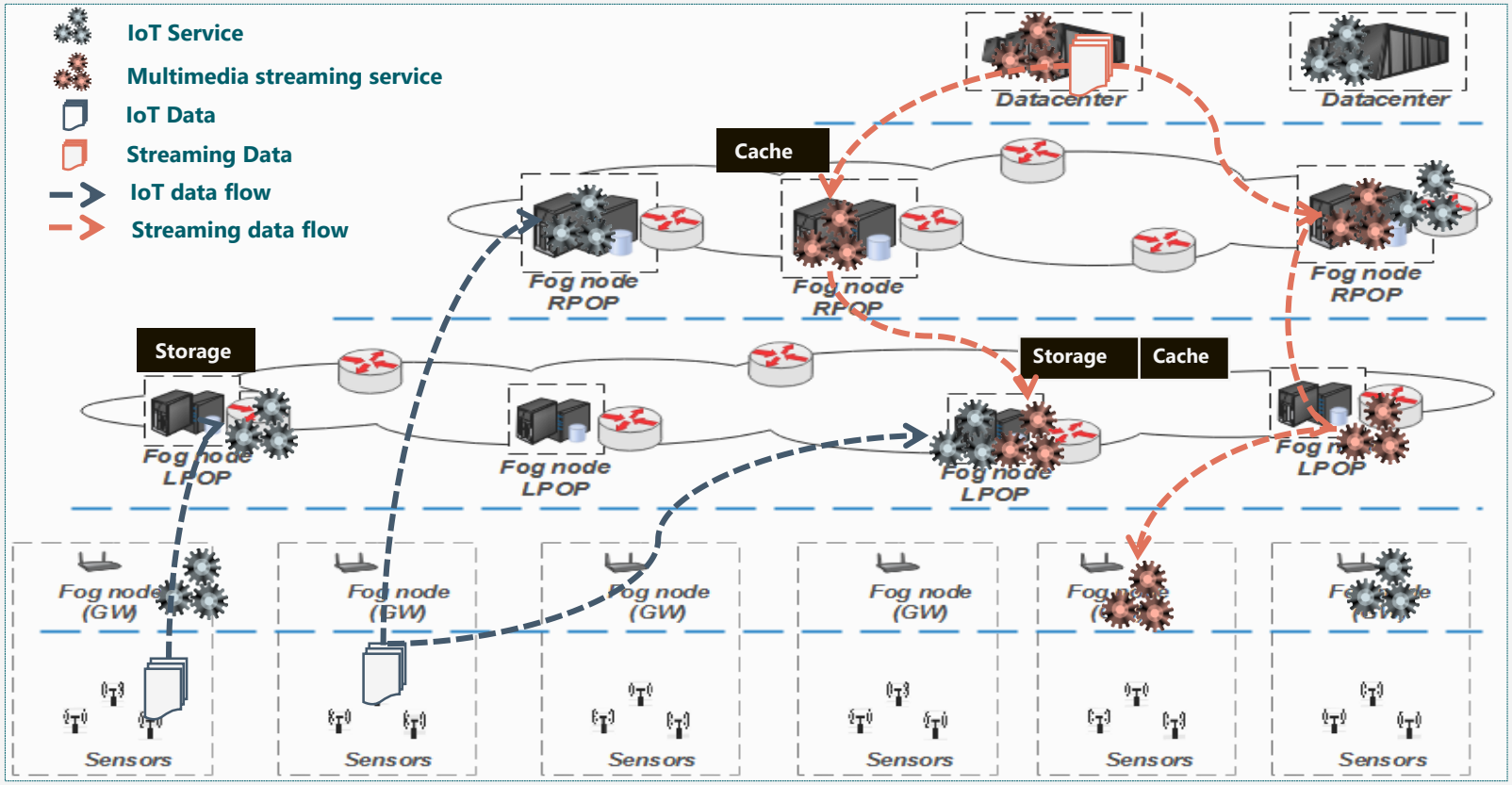
- Solving time



- Mohammed Islam Naas, Laurent Lemarchand, Philippe Raipin, Jalil Boukhobza. **IoT Data Replication and Consistency Management in Fog computing**. **Journal of Grid Computing**, Springer Verlag, 2021, 19 (3), pp.33

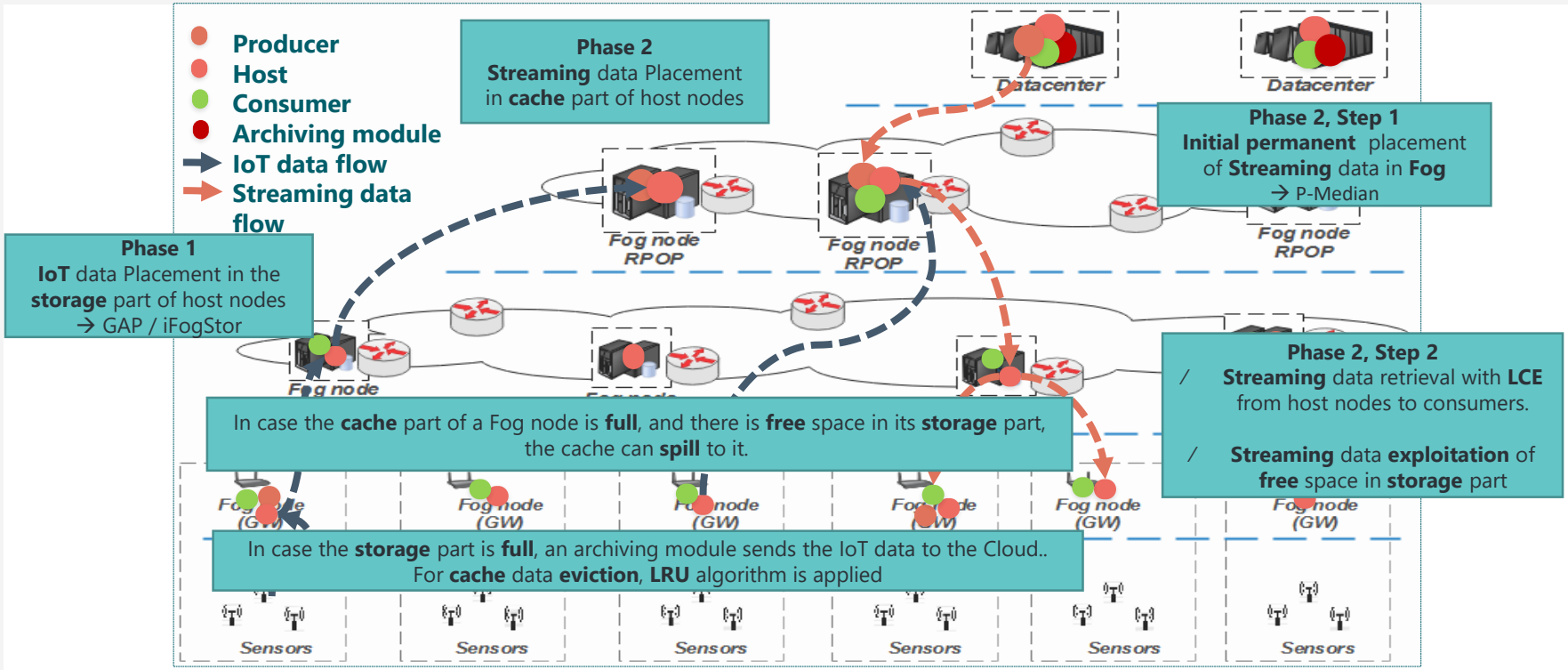
What if we mix workloads ?
IoT Data as in iFogStor and streaming data (that is most of the data on the internet …)





- Lydia Ait-Oucheggou, Mohammed Islam Naas, Yassine Hadjadj-Aoul, Jalil Boukhobza. **When IoT Data Meet Streaming in the Fog**. IEEE 6th International Conference on Fog and Edge Computing (**IEEE ICFEC**), May 2022, Messina, Italy. pp.50-57,

What if we mix workloads ?
IoT Data as in iFogStor and streaming data (that is most of the data …) → uFogStor (Unified stoarge …)



- Lydia Ait-Oucheggou, Mohammed Islam Naas, Yassine Hadjadj-Aoul, Jalil Boukhobza. **When IoT Data Meet Streaming in the Fog**. IEEE 6th International Conference on Fog and Edge Computing (**IEEE ICFEC**), May 2022, Messina, Italy. pp.50-57,

Presentation outline

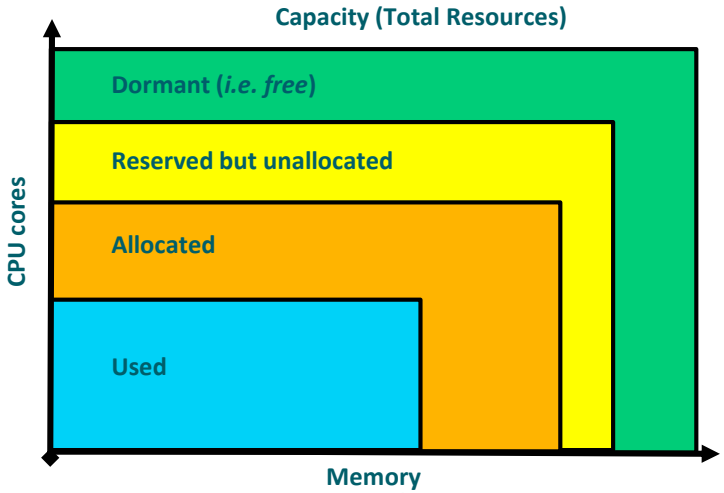
- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

How to make profit from Cloud unused resources ?

jalil.boukhobza@ensta-bretagne.fr

Context and motivation

75-65 % of the CPU and 50-60% of RAM cloud resources are underutilized [1]

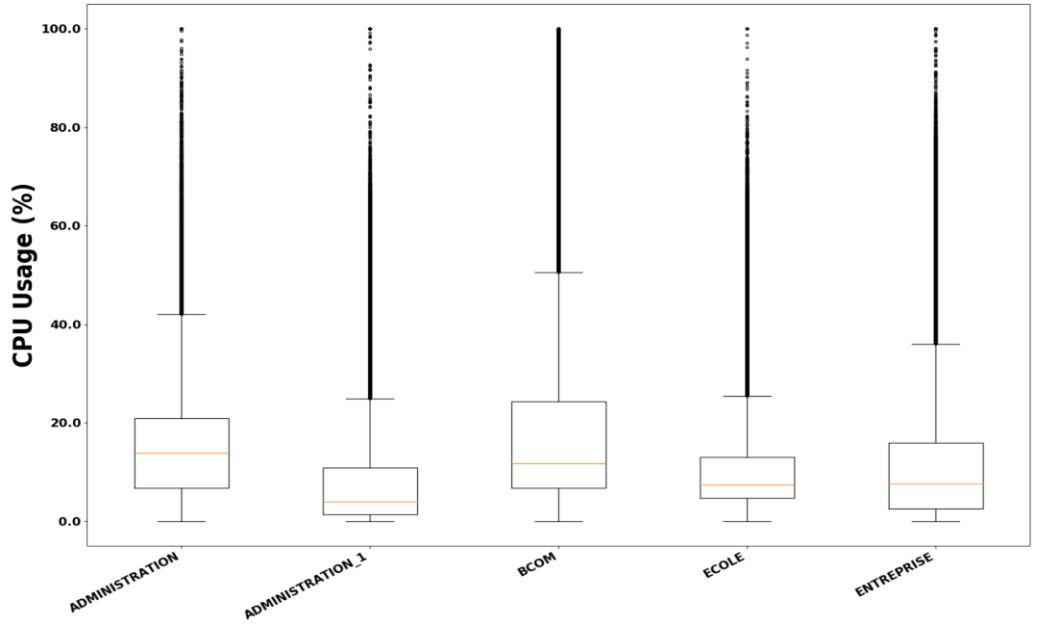


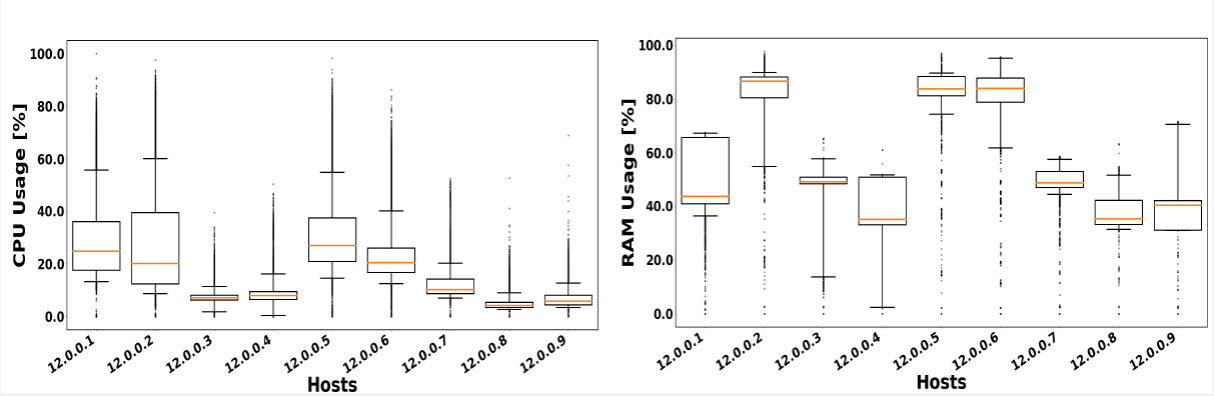
Opportunity: Optimize the cost of deploying applications by reclaiming Cloud unused resources

[1] Marcus Carvalho et al., "Long-term SLOs for reclaimed cloud computing resources", in: ACM Symposium on Cloud Computing (SoCC), Seattle, WA, USA, 2014

- Jean-Emile Dartois, **Leveraging Cloud unused heterogeneous resources for applications with SLA guarantees**, PhD thesis defended on Sept. 2020, https://theses.hal.science/tel-03009816

Context and motivation





Datasets

Name	History	Servers
ADMINISTRATION	35 months	7
ADMINISTRATION_1	13 months	70
BCOM	9 months	9
ECOLE	22 months	10
ENTREPRISE	17 months	27

- 51 metrics (e.g., CPU, RAM, I/O, Network)
- Frequency : Every minute
- 2017

- Jean-Emile Dartois, **Leveraging Cloud unused heterogeneous resources for applications with SLA guarantees**, PhD thesis defended on Sept. 2020, https://theses.hal.science/tel-03009816

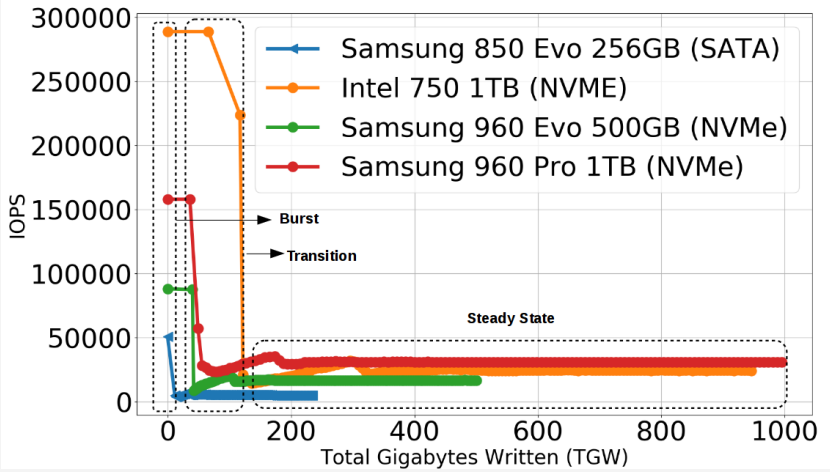
Our approach for such a problem

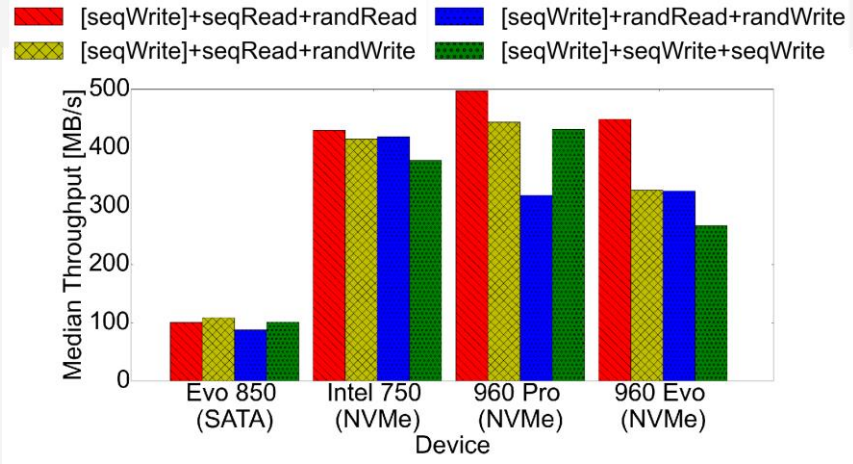
1. Evaluate the overall resource capacity
2. Evaluate resource utilization
3. Schedule jobs on top of ephemeral resources

Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

1) Evaluate the overall resource capacity





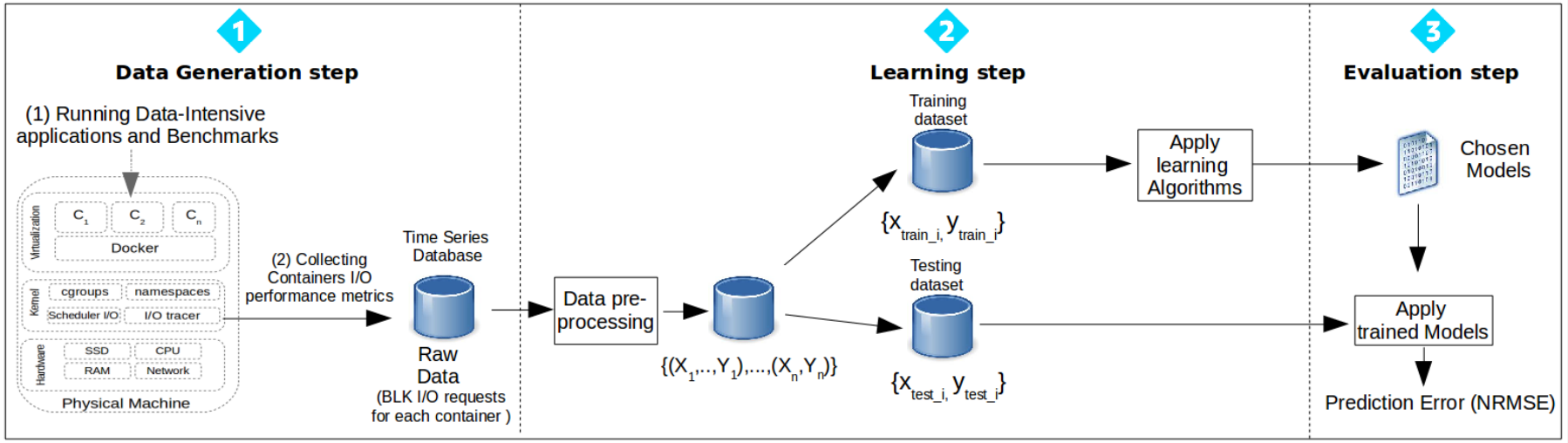
- ➔ Interference due to **SSD internal mechanisms** (e.g. GC,wear leveling)
- ➔ Interference due to **kernel I/O software stack** (e.g. page cache read-ahead and I/O scheduling)
- ➔ Interference due to **co-hosted applications workloads**

Can we model the throughput of an SSD knowing the executed applications ?

- Jean-Emile Dartois, Jalil Boukhobza, Anas Knefati, Olivier Barais, **Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization**, **IEEE Transactions on Cloud Computing**, vol. 9, issue 3., 1103-1116, 2021.

1) ML based method to model SSD capacity according to ran workloads

Evaluate capacity | Evaluate usage | Schedule tasks



- RQ1) What is the accuracy and the robustness of the tested algorithms?
- RQ2) How does the accuracy change with regards to the size of the training dataset (learning curve)?
- RQ3) What are the most important features in building the model?
- RQ4) What is the training time overhead?

Name	Category	Description
web	Server application	N-tiers web application
email	Server application	Email server
fileserver	Server application	File server
video	Multimedia processing	H.264 video transcoding
freqmine	Data mining	Frequent itemset mining
compile	Software development	Linux kernel compilation
micro-benchmark	Synthetic Benchmark	I/O workload generator

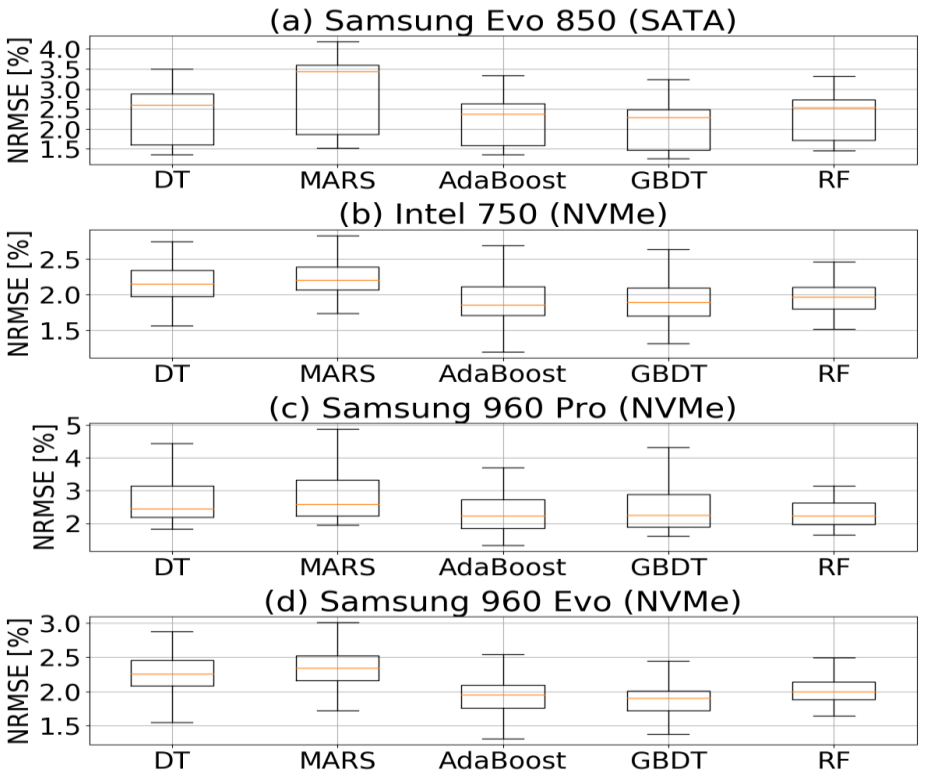
Key: ▲= good, ○=fair, and ▼=poor.

Characteristic	DT	MARS	AdaBoost	GBDT	RF
Robustness to outliers in input space	▲	▼	○	▲	▲
Handling of missing values	▲	▲	▲	▲	▲
complexity	▲	▲	▼	▼	▼
Prediction accuracy	▼	○	▲	▲	▲

T. H. R. T. J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics, 2013

- Jean-Emile Dartois, Jalil Boukhobza, Anas Knefati, Olivier Barais, **Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization, IEEE Transactions on Cloud Computing**, vol. 9, issue 3., 1103-1116, 2021.

1) Some results



RQ1 Prediction accuracy and models robustness:

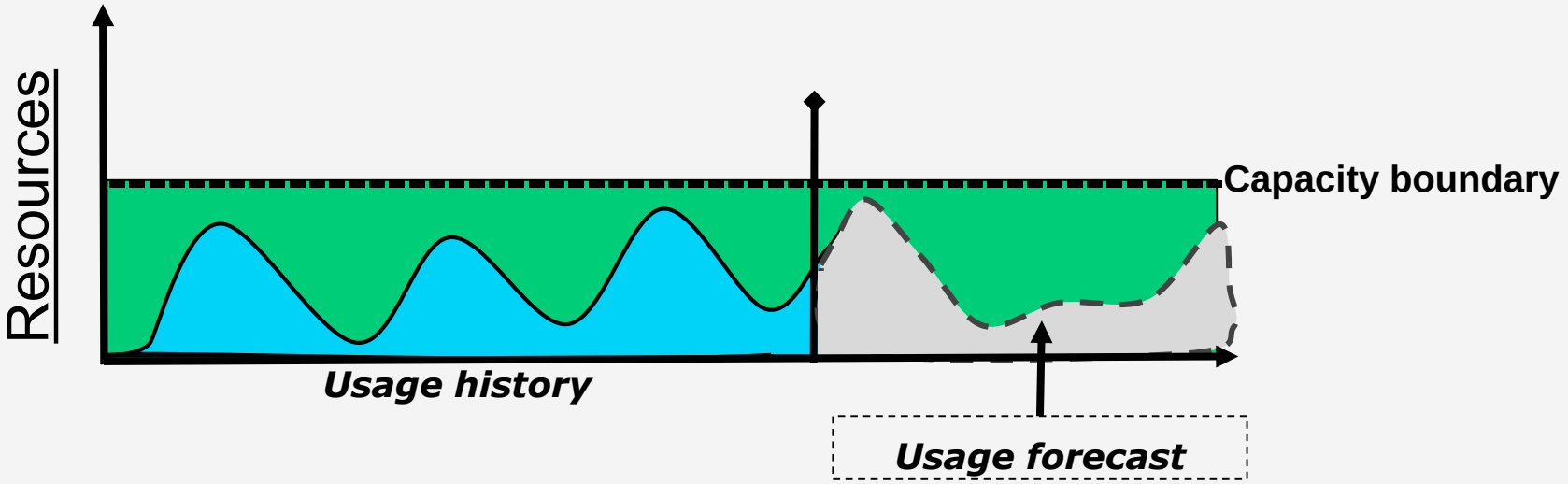
- The **ranking** of the tested algorithms was the **same** regardless of the **SSD** used
- GDBT, AdaBoost and RF gave the best accuracy with an **NRMSE of about 2.5%**
- Adaboost, GDBT and RF provided the **smallest dispersion** proving their robustness to a changing I/O
- We used **fixed hyperparameters** to tune RF and DT, this makes them simpler to use

- Jean-Emile Dartois, Jalil Boukhobza, Anas Knefati, Olivier Barais, **Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization, IEEE Transactions on Cloud Computing**, vol. 9, issue 3., 1103-1116, 2021.

Presentation outline

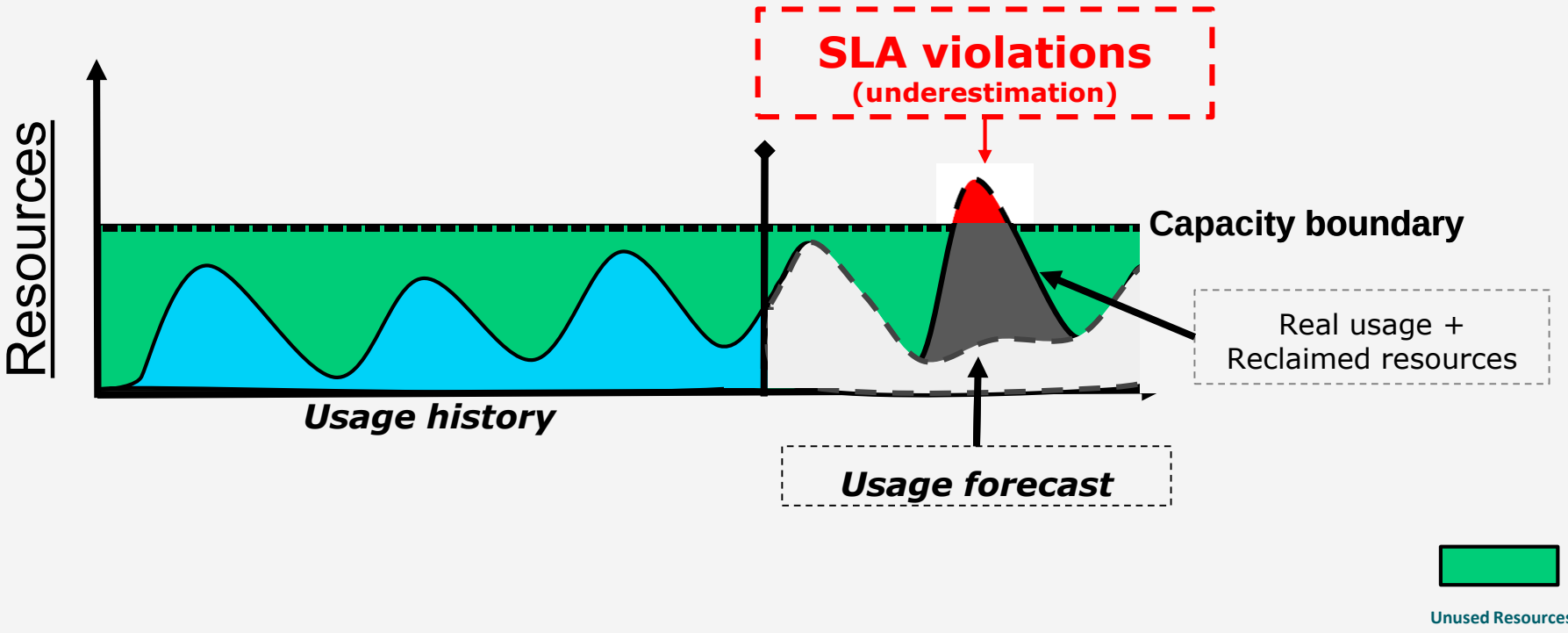
- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

2) Evaluate resource utilization



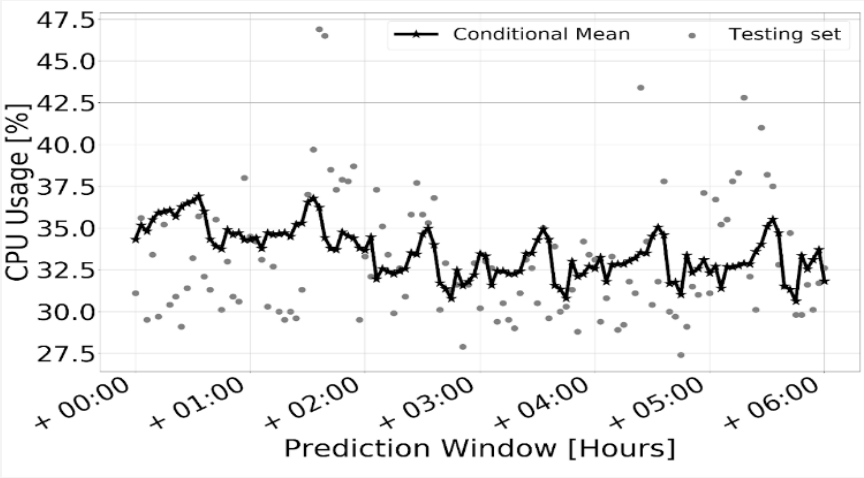
- Jean-Emile Dartois, Anas Knefati, Jalil Boukhobza, Olivier Barais, **Using Quantile Regression for Reclaiming UnusedCloud Resources while achieving SLA**, in proceedings of the 10th IEEE International Conference on Cloud Computing Technology and Science (**IEEE CloudCom**), pp. 89-98, Nicosia, December 2018

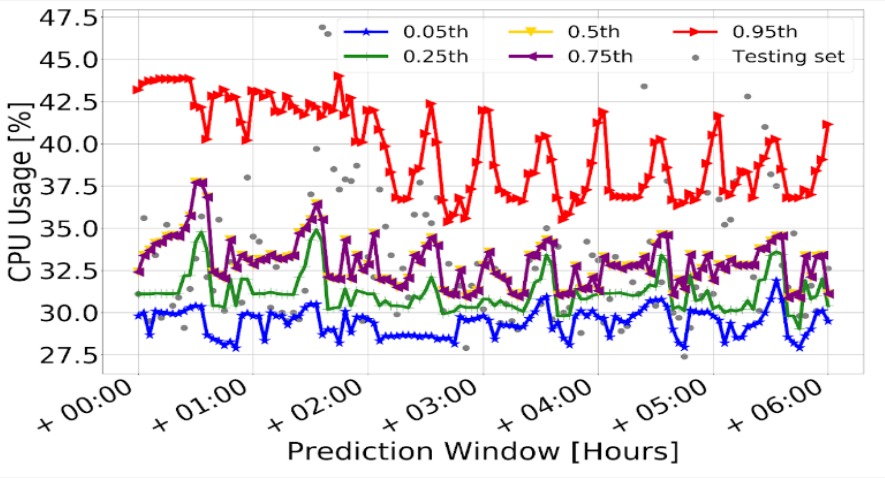
2) Evaluate resource utilization



- Jean-Emile Dartois, Anas Knefati, Jalil Boukhobza, Olivier Barais, **Using Quantile Regression for Reclaiming UnusedCloud Resources while achieving SLA**, in proceedings of the 10th IEEE International Conference on Cloud Computing Technology and Science (**IEEE CloudCom**), pp. 89-98, Nicosia, December 2018

2) Using quantile regression for resource usage prediction



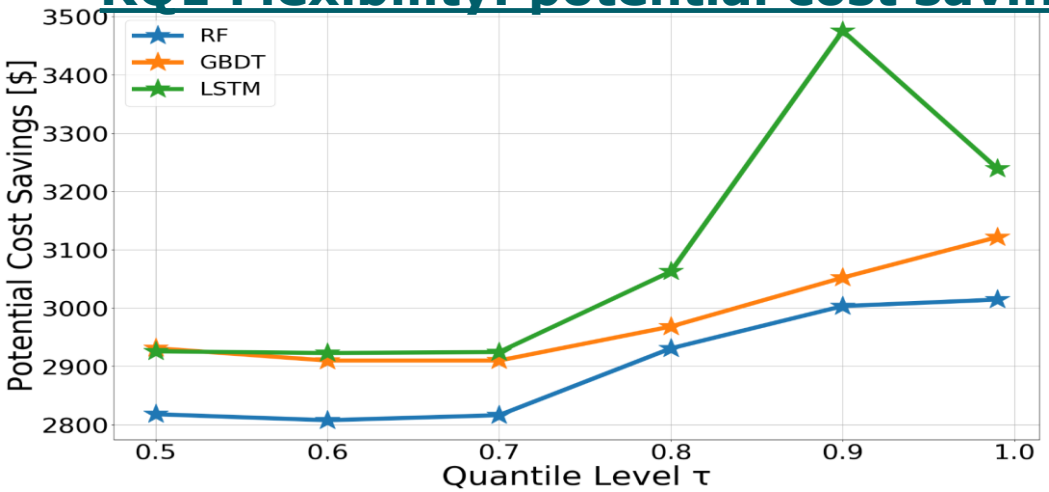


- Evaluated: GBDT, RF, LSTM
- Some important criteria:
  - **Granularity**: level at which the estimation is performed
  - **Flexibility**: trade-off between the amount of resources to reclaim and the risk of SLA violations
  - **Exhaustivity**: several resource metrics to achieve SLA requirements
  - **Robustness**: robust to workload change
  - **Applicability**: low overhead

- Jean-Emile Dartois, Anas Knefati, Jalil Boukhobza, Olivier Barais, **Using Quantile Regression for Reclaiming UnusedCloud Resources while achieving SLA**, in proceedings of the 10th IEEE International Conference on Cloud Computing Technology and Science (**IEEE CloudCom**), pp. 89-98, Nicosia, December 2018

2) Some results

RQ1 Flexibility: potential cost savings



- All learning algorithms:
  - **Increase potential cost savings** with the increase of τ
  - **Increase up to 20%** cost savings compared to **median-estimation** based approach (τ=0.5)
- When τ > 0.9 the reduction of unused resources is higher than the decrease of SLA violations

- Jean-Emile Dartois, Anas Knefati, Jalil Boukhobza, Olivier Barais, **Using Quantile Regression for Reclaiming UnusedCloud Resources while achieving SLA**, in proceedings of the 10th IEEE International Conference on Cloud Computing Technology and Science (**IEEE CloudCom**), pp. 89-98, Nicosia, December 2018

Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

3) Schedule jobs on top of ephemeral resources
a) Cuckoo: running BigData jobs on ephemeral Cloud resources

Evaluate capacity → Evaluate usage → Schedule tasks

Use of free resources

Avoid interference / SLA violation

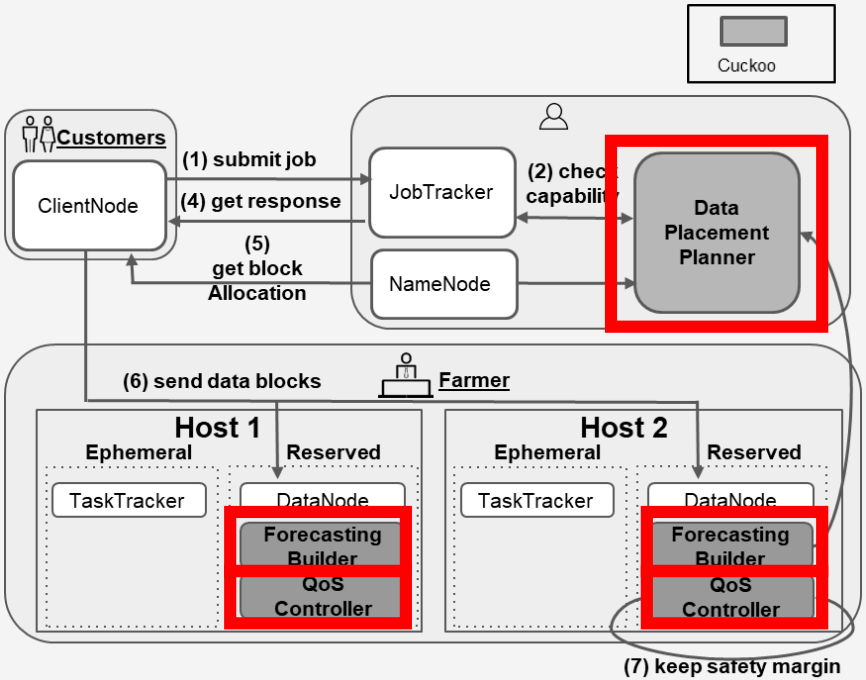
Customers SLA satisfaction

- Jean-Emile Dartois, Heverson Ribeiro, Jalil Boukhobza, Olivier Barais, **Cuckoo: a Mechanism for Exploiting Ephemeral and Heterogeneous Cloud Resources**, accepted in the IEEE International Conference on Cloud Computing (**IEEE CLOUD**), Milano, 2019

3) Schedule jobs on top of ephemeral resources
a) Cuckoo: running BigData jobs on ephemeral Cloud resources

Evaluate capacity → Evaluate usage → **Schedule tasks**

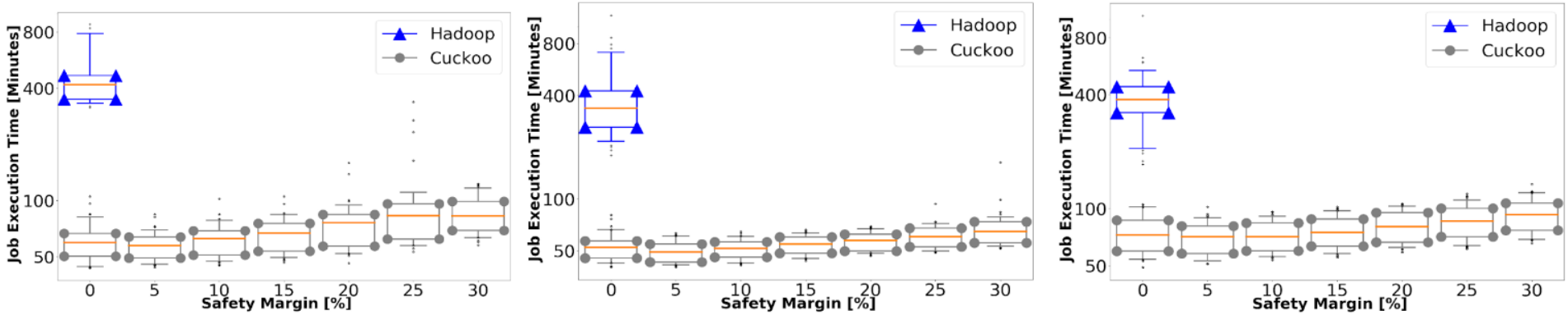
- A framework to leverage unused resources to run MapReduce jobs
  - **Forcasting builder** → predicts host resource utilization
  - **Data placement planner** →
    - distribute data chunks based on the forecasting builder / uses a **weighted round robin algorithm (based on processing capacity + volatility)**
    - For tasks: uses the default Hadoop scheduler
  - **QoS controller**: guarantees there is no interference with regular workload → continuous monitoring
    - Uses a **(static) safety margin** on unused resources



- Jean-Emile Dartois, Heverson Ribeiro, Jalil Boukhobza, Olivier Barais, **Cuckoo: a Mechanism for Exploiting Ephemeral and Heterogeneous Cloud Resources**, accepted in the IEEE International Conference on Cloud Computing (**IEEE CLOUD**), Milano, 2019

3) Schedule jobs on top of ephemeral resources

a) Cuckoo: running BigData jobs on ephemeral Cloud resources – some results



(a) **Private Company 1** (b) **Private Company 2** (c) **University**

Figure 2. Job Execution Time for standard Hadoop and Cuckoo

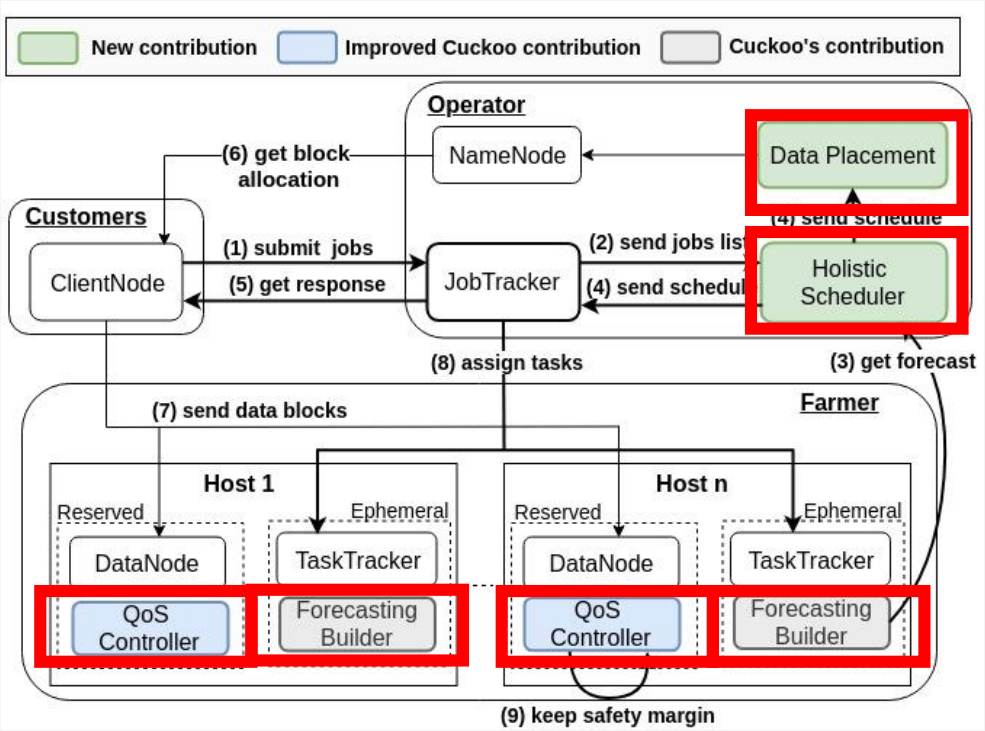
- **Smaller dispersion** and **best completion time** (**5% safety margin**)
- Cuckoo is **7 times faster** than native Hadoop strategy for PC-1 and PC-2 and 5 **times faster** for the University with a safety margin of 5%
- See the paper for more results (on remote execution + relaunched tasks)

- Jean-Emile Dartois, Heverson Ribeiro, Jalil Boukhobza, Olivier Barais**, Cuckoo: a Mechanism for Exploiting Ephemeral and Heterogeneous Cloud Resources**, accepted in the IEEE International Conference on Cloud Computing (**IEEE CLOUD**), Milano, 2019

3) Schedule jobs on top of ephemeral resources
b) Salamander: a Holistic Scheduling of MapReduce Jobs on Ephemeral Cloud Resources

Evaluate capacity → Evaluate usage → Schedule tasks

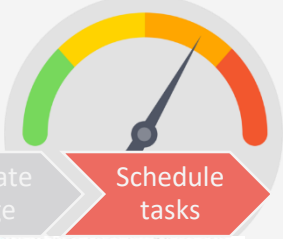
- Forecasting builder
- **Holistic scheduler**
  - Three solving strategies to schedule MapReduce jobs and tasks
  - **(1) Constraint Prog., (2) Genetic** and **(3) Local Search-based algorithms**
- **Data placement**: data follow tasks for placement
- **QoS controller**:
  - Compressible resources → CPU: throttle
  - Incompressible ones → RAM: relaunch



- Mohamed Handaoui, Jean-Emile Dartois, Laurent Lemarchand, Jalil Boukhobza, **Salamander: a Holistic Scheduling of MapReduce Jobs on Ephemeral Cloud Resources**, In Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (**IEEE/ACM CCGRID**), May 2020

3) Schedule jobs on top of ephemeral resources

b) Salamander: a Holistic Scheduling of MapReduce Jobs on Ephemeral Cloud Resources



- Optimization problem
- Amount of unused resources:
  - $A(k, m, t) = C(k, m) * (1 - (U(k, m, t) + S(m)))$
- Objective functions:

$$maximize(\sum_{i \in J} p(i))$$

Nbr of scheduled tasks

$$minimize(\max_{i \in J}(ts(i) + Te(i, n(i))))$$

Sum of tasks makespan

- Constraints

$$\forall i \in J \ : \ 0 \leq ts(i) + Te(i, n(i)) \leq T$$

Bound of execution time

$$\forall i, j \in J \mid D(i, j) = 1 \ : \ (n(i) \geq 0 \vee n(j) \geq 0) \implies (n(i) \geq 0 \wedge n(j) \geq 0) \wedge (ts(j) \geq ts(i) + Te(i, n(i)))$$

a job has to be either fully scheduled or rejected

$$\forall t \leq T, \forall k \in N, \forall m \in R \ : \ \left(\sum_{\substack{\{i \mid (i \in J) \wedge (n(i) = k) \wedge \\ (0 \leq t - ts(i) \leq Te(i, n(i)))\}}} Rq(i, m) \right) \leq A(k, m, t)$$

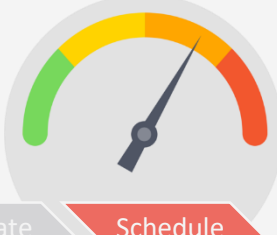
A task is scheduled if all required resources are available

Input variables: uppercase letters. **Output variables**: lowercase letters.

Type	Notation	Domain	Description
Input	N	-	List of available nodes' IDs
	J	-	List of MapReduce jobs with tasks' IDs
	R	-	List of resource metrics (*e.g.*, CPU, RAM)
	T	$\mathbb{N}$	Size of the scheduling window (*e.g.*, 24-hours with 3 minutes sampling = 480 points)
	$C(k, m)$	$\mathbb{N}$	Maximum capacity given a resource metric m in a node k
	$U(k, m, t)$	$[0, 1]$	Predicted utilization of resource metric m in a node k at time t
	$S(m)$	$[0, 1]$	Safety margin percentage for resource metric m
	$A(k, m, t)$	$\mathbb{N}$	Amount of available resources of metric m in a node k at time t
	$Te(i, n(i))$	$\mathbb{N}$	Estimated execution time of task i, it varies depending on the computational capacities of the assigned node $n(i)$
	$Rq(i, m)$	$\mathbb{N}$	Requirement of task i in terms of resource metric m
	$D(i, j)$	$\{0, 1\}$	Presence of a dependency between task i and j
Output	$ts(i)$	T	Start time of task i
	$n(i)$	N	The node ID that the task i is assigned to
	$p(i)$	$\{0, 1\}$	Presence of task i in the schedule, inferred from the start time $ts(i)$ or the task's node $n(i)$

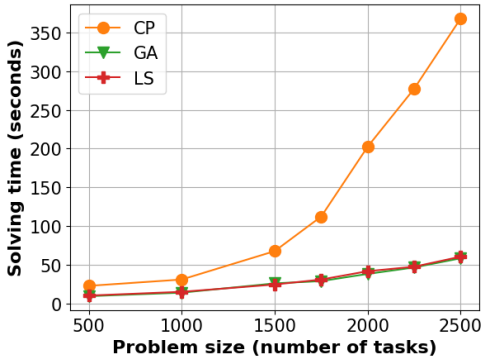
- Mohamed Handaoui, Jean-Emile Dartois, Laurent Lemarchand, Jalil Boukhobza, **Salamander: a Holistic Scheduling of MapReduce Jobs on Ephemeral Cloud Resources**, In Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (**IEEE/ACM CCGRID**), May 2020

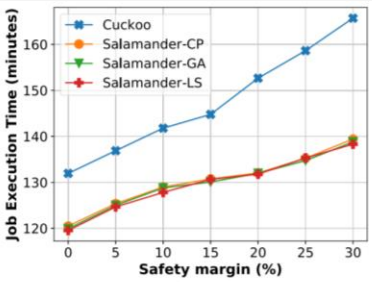
3) Schedule jobs on top of ephemeral resources
b) Salamander: a Holistic Scheduling of MapReduce Jobs on Ephemeral Cloud Resources



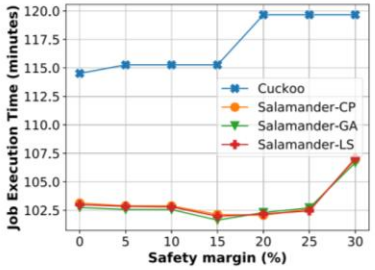
Evaluate capacity | Evaluate usage | Schedule tasks

- Comparable performance between different solving algorithms
- Always better than Cuckoo + more realistic as it places tasks (considering dependencies) and not only data
- Solving times
  - GA and LS showed to be equivalent
  - CP scales poorly but is (a bit) better in terms of makespan

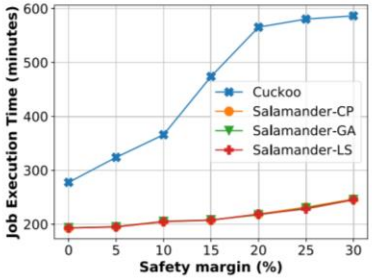




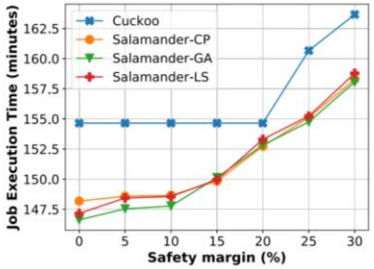
(a) PC-1 - 128 MB



(b) PC-2 - 128 MB



(d) PC-1 - 256 MB



(e) PC-2 - 256 MB

- Mohamed Handaoui, Jean-Emile Dartois, Laurent Lemarchand, Jalil Boukhobza, **Salamander: a Holistic Scheduling of MapReduce Jobs on Ephemeral Cloud** Resources, In Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (**IEEE/ACM CCGRID**), May 2020

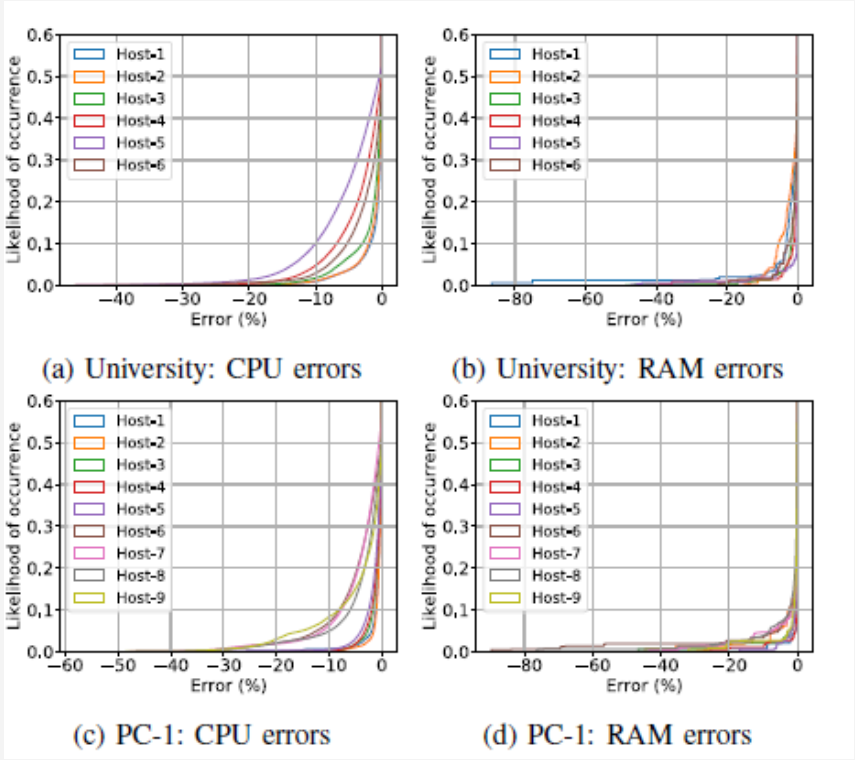
3) Schedule jobs on top of ephemeral resources

c) Releaser: A Reinforcement Learning Strategy for Optimizing Utilization Of Ephemeral Cloud Resources

Evaluate capacity → Evaluate usage → Schedule tasks

- **Motivation**: variability in the precision of resource estimation.
- **Idea of Releaser:** adapt the safety margin accordingly
  - If resource estimation is sure → lower the safety margin
  - If it is likely to be imprecise → increase the safety margin
- **Technique used:** Use of RL to solve the problem using Deep Deterministic Policy Gradient (DDPG) to maximize:

$csavings(h, d) = cpotential\ saving(h, d) - cpenalty(h, d)$

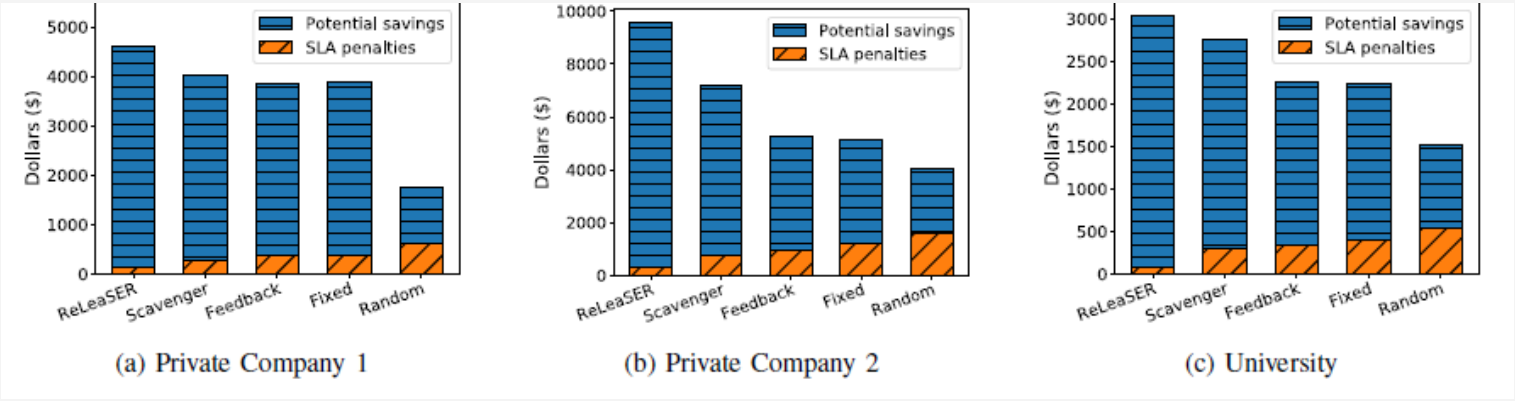


- Mohamed Handaoui, Jean-Emile Dartois, Jalil Boukhobza, Olivier Barais, Laurent d'Orazio. **ReLeaSER: A Reinforcement Learning Strategy for Optimizing Utilization Of Ephemeral Cloud Resources**. in Proceedings of the 2th IEEE International Conference on Cloud Computing Technology and Science (**IEEE CloudCom**), Dec 2020, Bangkok, Thailand

3) Schedule jobs on top of ephemeral resources

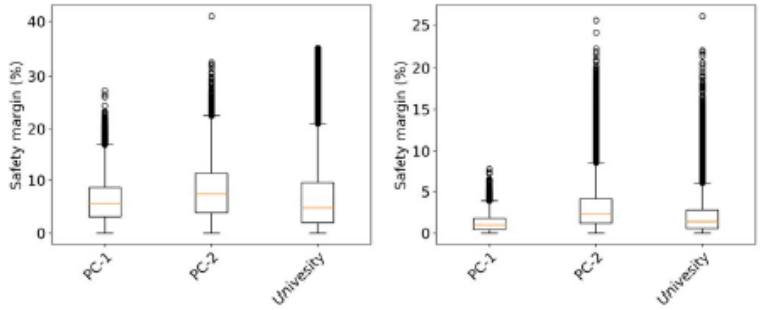
c) Releaser: A Reinforcement Learning Strategy for Optimizing Utilization Of Ephemeral Cloud Resources

Evaluate capacity → Evaluate usage → Schedule tasks



(a) Private Company 1 (b) Private Company 2 (c) University

- Adpting safety margin helps using more efficiently the available resources → generating more profit
- Adapting the safety margin seems to be relevant as the predictions accuracy depend of the host



(a) CPU safety margins (b) RAM safety margins

- Mohamed Handaoui, Jean-Emile Dartois, Jalil Boukhobza, Olivier Barais, Laurent d'Orazio. **ReLeaSER: A Reinforcement Learning Strategy for Optimizing Utilization Of Ephemeral Cloud Resources**. in Proceedings of the 2th IEEE International Conference on Cloud Computing Technology and Science **(IEEE CloudCom)**, Dec 2020, Bangkok, Thailand

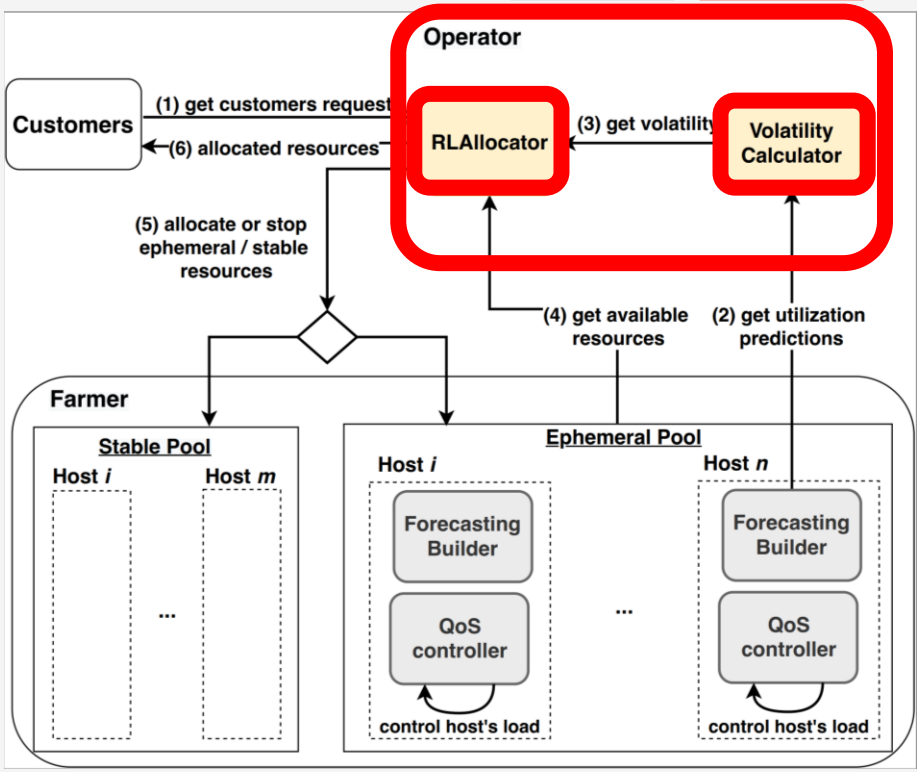
3) Schedule jobs on top of ephemeral resources

d) Riscless: A ReInforcement Learning Strategy to guarantee SLA on CLoud Ephemeral and Stable ReSources

Evaluate capacity → Evaluate usage → Schedule tasks

- Avoiding the risk of safety margin bad dimensionning → **No safety margin**
- **Riscless idea:** use a minimal set of stable resources to absorb ephemeral resource volatility at a minimal cost while guaranteeing SLA
- Modules
  - **Forecating builder**: prediction of 24h of resource utilization
  - **QoS controller**: SLA guarantee for regular customers
  - **Volatility calculator**: synthesizing information of the Forecating builder → volatility rate
  - **RLAllocator**: decision maker → when and how much resources (ephemeral + stable) to allocate

Contribution of this paper



Reward function → **Maximize ephemeral resource usage** / **reduce the use of stable resources** / reduce SLA violations

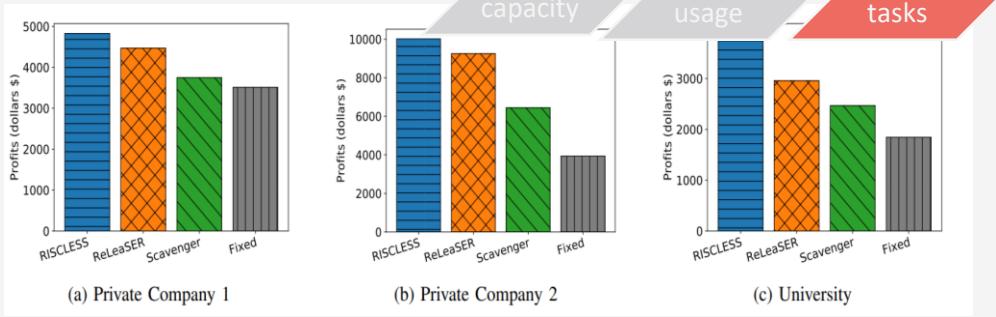
- Sidahmed Yalles, Mohamed Handaoui, Jean-Emile Dartois, Olivier Barais, Laurent d'Orazio, Jalil Boukhboza, **RISCLESS: A Reinforcement Learning Strategy to Guarantee SLA on Cloud Ephemeral and Stable Resources**. 2022 - 30th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (**Euromicro PDP**), Mar 2022, Valladolid, Spain. pp.83-87

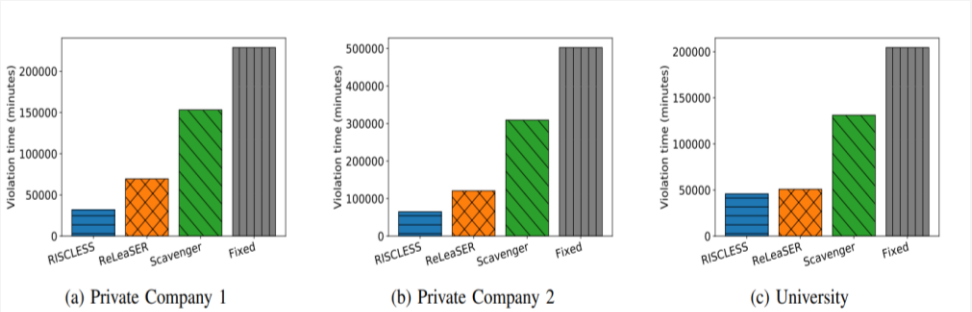
3) Schedule jobs on top of ephemeral resources

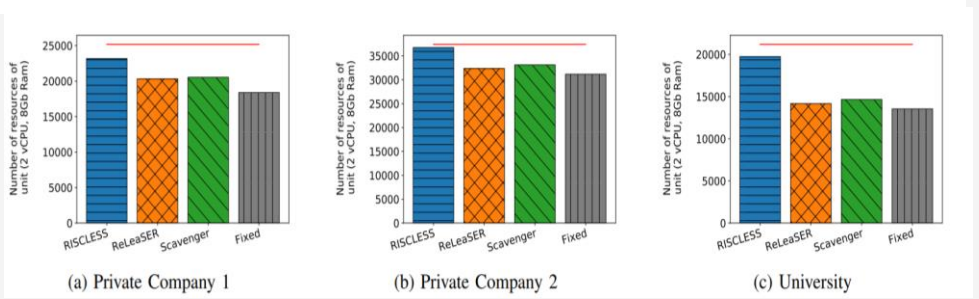
d) Riscless: A ReInforcement Learning Strategy to guarantee SLA on CLoud Ephemeral and Stable ReSources

Evaluate capacity → Evaluate usage → Schedule tasks

- Total profites: RISCLESS > ReLeaSer > Scavenger > Fixed
  - SLA violation and amount of reclaimed resources
- SLA violation: RISCLESS < ReLeaSer < Scavenger < Fixed
  - Using stable resources may decrease SLA violations
- Amout of ephemeral resources used:
  - Redline is the total available
  - RISCLESS is the most approaching strategy.







- Sidahmed Yalles, Mohamed Handaoui, Jean-Emile Dartois, Olivier Barais, Laurent d'Orazio, Jalil Boukhboza, **RISCLESS: A Reinforcement Learning Strategy to Guarantee SLA on Cloud Ephemeral and Stable Resources**. 2022 - 30th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (**Euromicro PDP),** Mar 2022, Valladolid, Spain. pp.83-87

3) Schedule jobs on top of ephemeral resources
Next step → scheduling on top of heterogeneous resources

Resources Allocation in Heterogeneous Serverless Cloud

An Application to Deepfake Detection

Vincent Lannurien,*† Laurent D'Orazio,*‡ Olivier Barais*‡
Esther Bernard,* Olivier Weppe,* Laurent Beaulieu,* Amine Kacete*
Stéphane Paquelet,* Jalil Boukhobza*†

December 4, 2022

* b<>com Institute of Research and Technology
† ENSTA Bretagne, Lab-STICC, CNRS, UMR 6285
‡ Univ. Rennes, Inria, CNRS, IRISA

This morning talk !!

Presentation outline

- Background on memory & storage
- Data placement in the Cloud/Edge
  - MAPE-K
  - Tracing I/Os
  - Analyzing I/Os
  - Planning for I/Os
  - Executing I/Os
- Ephemeral resource management in the Cloud
  - Capacity
  - Usage
  - Scheduling
- Some conclusions

Conclusions

- Short conclusion
- Long conclusion

jalil.boukhobza@ensta-bretagne.fr

Conclusion: more to come on NVM …

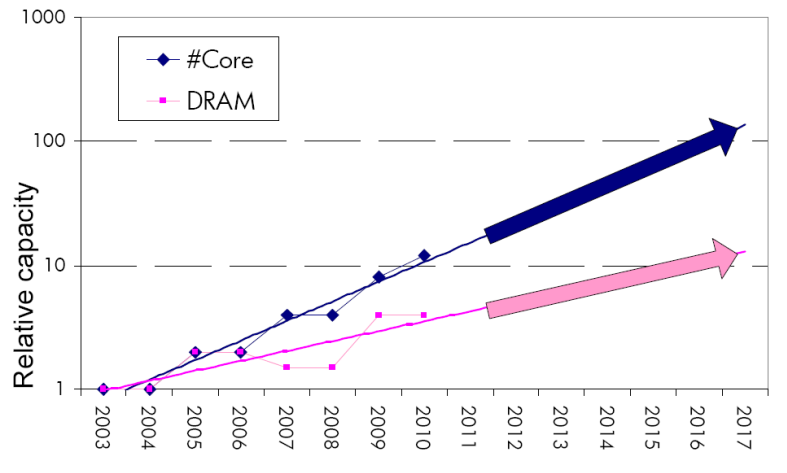
- Non-Volatile-Memory, emerged during the last 2 decades
- Definition: Solid state memory (no moving parts) that do not need to have their memory contents periodically refreshed (source : http://searchstorage.techtarget.com/definition/nonvolatile-memory)
- Flash memory, Phase Change memory (PRAM or PCM), Resistive Memory (ReRAM), Magneto-resistive Memory (STT-RAM), Ferroelectric memories (FeRAM) …

- **Why ?,** mainly because of …
  - DRAM scaling
  - Energy consumption

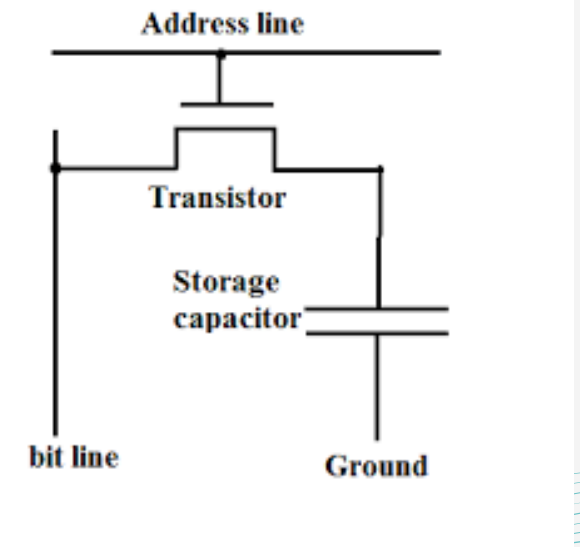
DRAM scaling issue

- Stores the charge in a capacitor
  - Size of capacitor matters → large for a reliable sensing
  - Size of transistor matters → large to reduce leakage and increase retention time
  - Scaling is difficult (ITRS)

Core count doubling ~ every 2 years
DRAM DIMM capacity doubling ~ every 3 years



Source: Onur Mutlu



Charge vs Resistive memory

- Charge memory
  - Write operation: capture charge Q
  - Read operation: detect the voltage V
  - Example: DRAM, flash memory

- Resistive memory
  - Write operation: pulse current dQ/dt
  - Read operation: detect the resistance R
  - Example: PCM, STT-RAM, ReRAM

jalil.boukhobza@ensta-bretagne.fr

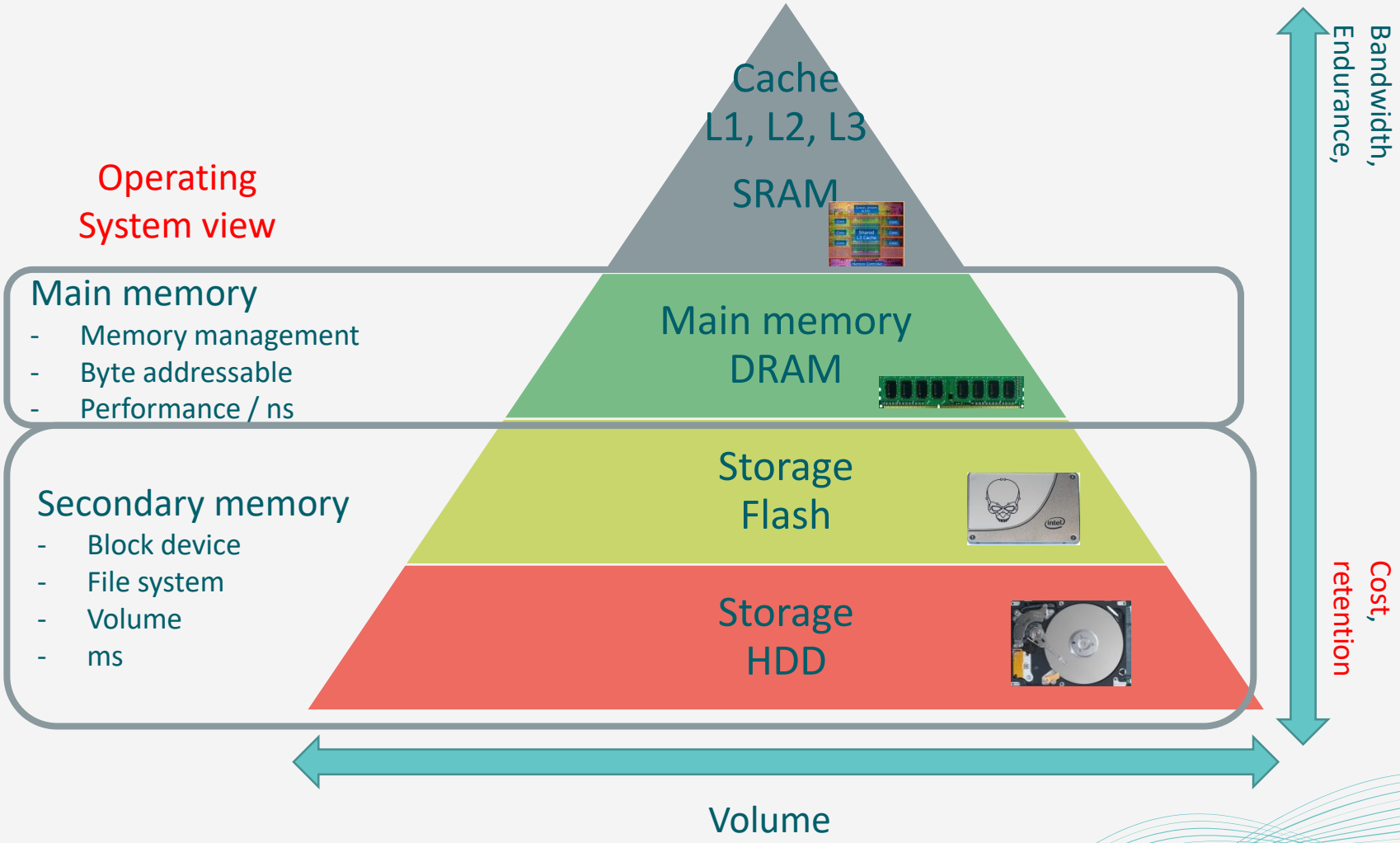
NVM: why is it so attractive ?

* Jalil Boukhobza, Stéphane Rubini, Renhai Chen, Zili Shao, **Emerging NVM: A Survey on Architectural Integration and Research Challenges**, **ACM Transactions on Design Automation of Electronic Systems (TODAES),** 23(2), 14:1-14:32 (2018).

	SRAM	DRAM	HDD	NAND flash	STT-RAM	ReRAM	PCM
Cell size (F^2)	120-200	6-10	N/A	4-6	6-50	4-10	4-12
Write endurance	10^{16}	$>10^{15}$	$>10^{15}$ (pb: mechanical parts)	10^4-10^5	**10^{12}-10^{15}**	**10^8-10^{11}**	**10^8-10^9**
Read Latency	~0.2-2ns	~10ns	3-5ms	15-35 µs	**2-35ns**	**~10ns**	**20-60ns**
Write Latency	~0.2-2ns	~10ns	3-5ms	200-500µs	**3-50ns**	**~50ns**	**20-150ns**
Leakage Power	High	Medium	(mechanical parts)	Low	**Low**	**Low**	**Low**
Dynamic Energy (R/W)	Low	Medium	(mechanical parts)	Low	**Low/High**	**Low/High**	**Medium/High**
Maturity	Mature	Mature	Mature	Mature	Manufactured	Test chips	Manufactured

Sources: [Vetter15] [Mittal15] [Xia15] [Wang'14] J.[Suresh14] [Baek13] [Maena15]

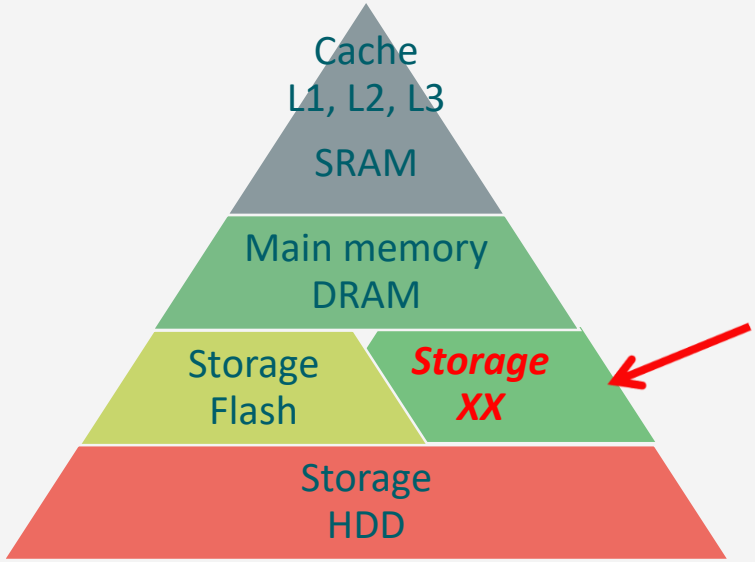
NVM Integration

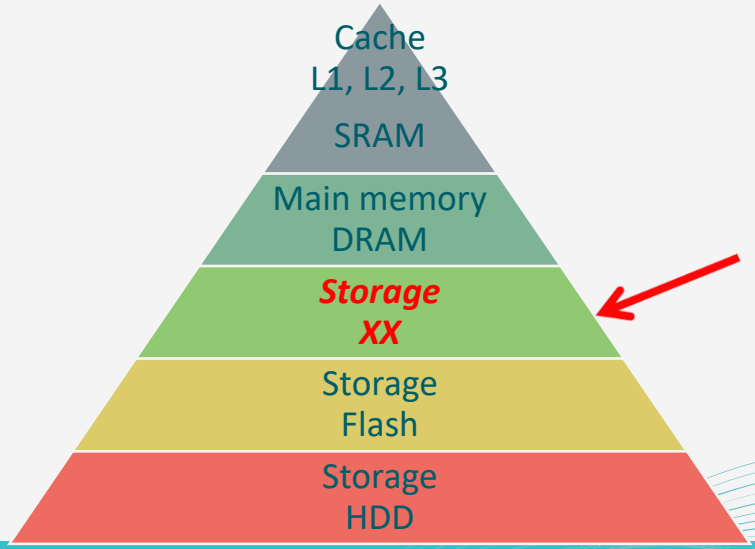


Horizontal and Vertical integration

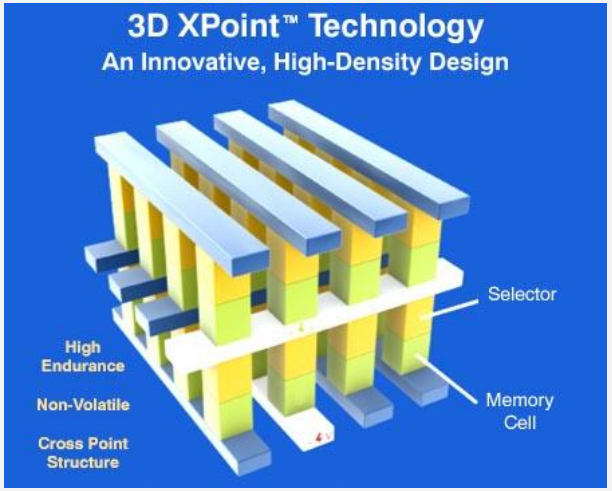
- Horizontal integration
  - Same interface as an existing memory
  - Data placement (controller, OS, …)

- Vertical integration
  - Different interface
  - Cache subsystem



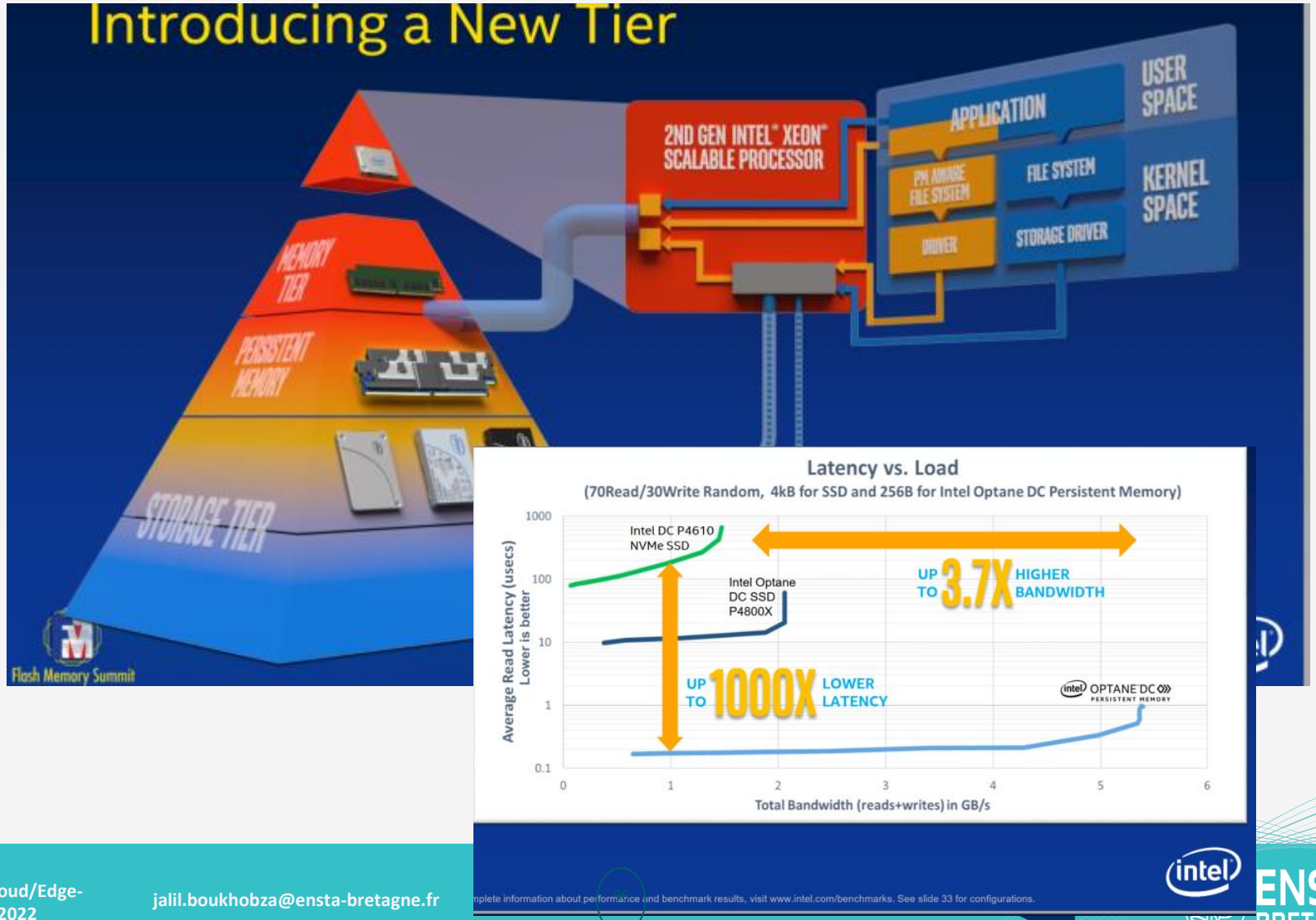


The case of 3D Xpoint



- Announced by Micron/Intel in 2015
- Intel Optane Memory Media
- PCM technology, 20nm
- Integrated in fast NVMe SSD and DIMM modules
- More Optane 3D Xpoint bits sold than all other emerging memories combined in 2019
- 2nd generation in 2020
- For memory integration: not supposed to replace DRAM, it supplements it (DRAM invisible to application, verticaland horizontal integration)
  - ~Xpoint: DRAM → 5:1 (Intel recommendation)

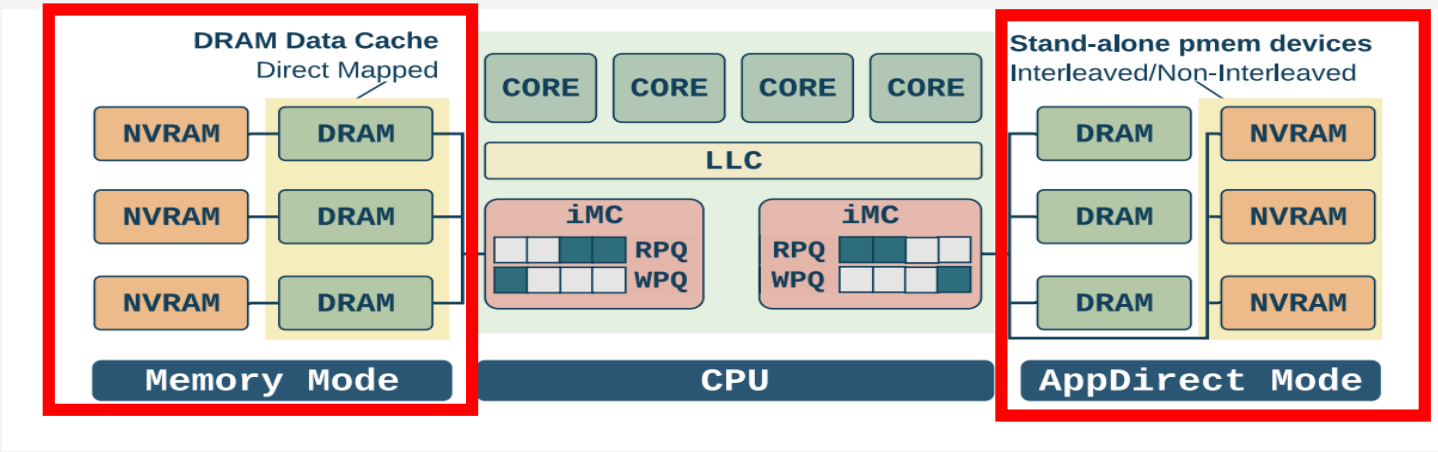
Performance and integration



jalil.boukhobza@ensta-bretagne.fr

Intel Optane DC architecture and performance figures

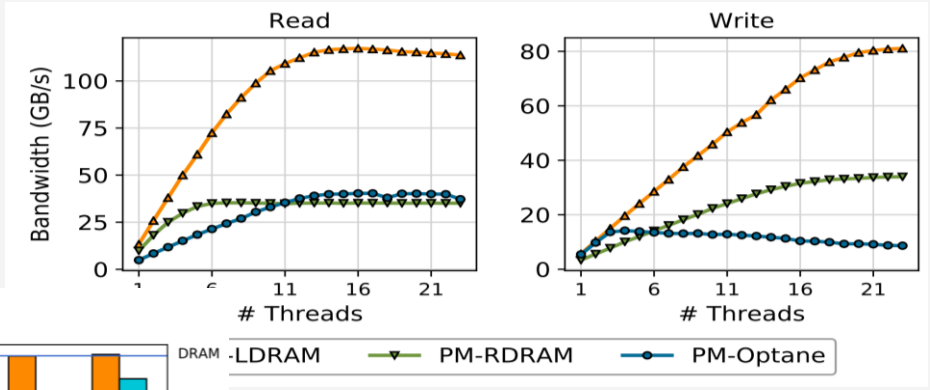
- 2 modes
  - Memory mode
    - Vertical integration with DRAM being a cache to NVRAM
    - → for scalability
  - AppDirect mode
    - Horizontal integration
    - → for persistance (performance compared to traditional storage) + bye addressability (execute in-place)

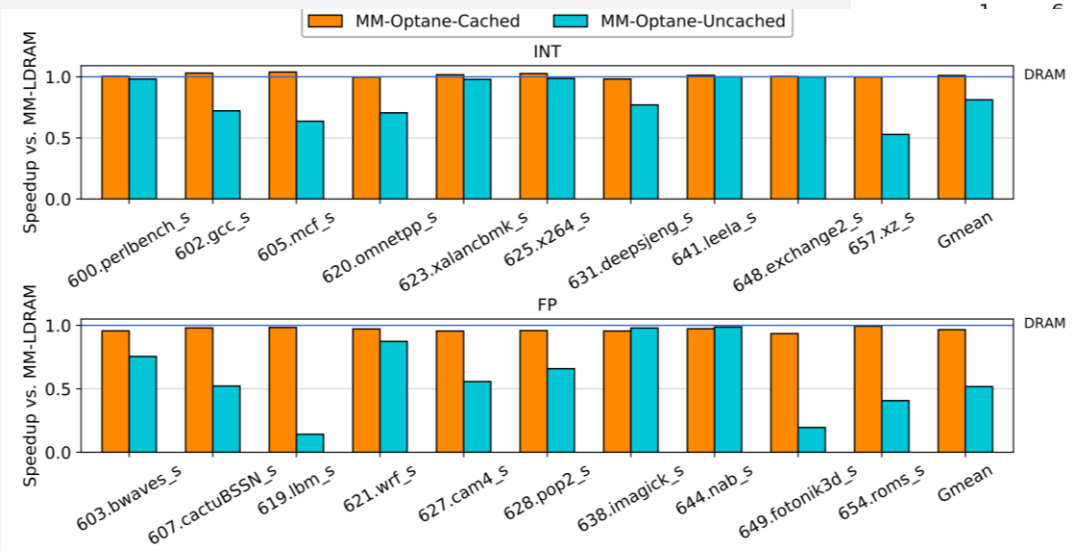


Zixuan Wang, Xiao Liu, Jian Yang, Theodore Michailidis, Steven Swanson, Jishen Zhao, Characterizing and Modeling Non-Volatile Memory Systems, IEEE Micro 2020.

Intel Optane DC architecture and performance figures -2-

- Latency
  - Reads 2 to 3x slower than DRAM (~80ns → ~300ns)
  - Writes same in latency (unless device saturated) → cached writes (~86ns → 94ns)
- Bandwidth
  - AppDirect mode (NVRAM)
- What about real apps ?





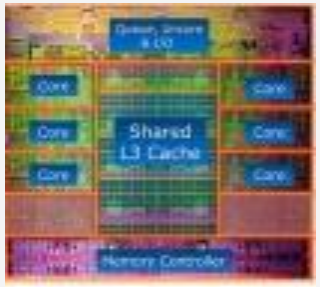
J. Izraelevitz, J. Yang, L. Zhang, J. Kim, X. Liu, A. Memaripour, Y. Joon Soh, Z. Wang, Y. Xu, S. R. Dulloor, J. Zhao, S. Swanson, Basic Performance Measurements of the Intel Optane DC Persistent Memory, 2019Module https://arxiv.org/pdf/1903.05714.pdf

NVM characteristics

- Common characteristics
  - Byte addressable
  - Performance properties → very good for read, good for write
  - Energy properties → static power ↘, dynamic power ?
  - Scalability
- Constraints to deal with
  - Performance asymmetry
    - Between write and read
    - Writing 1 ≠ writing 0
  - Energy consumption asymmetry
  - Wear out

NVM takeaway

- **You cannot avoid it !**
- **As a processor cache (mainly STT-RAM)**
  - High frequency access → low latency and high endurance
    - NVM ☹
  - Last level cache → acceptable
    - Many write operations → wear leveling
  - Technology compatibility



- **As a main memory (STT-RAM, PCM & ReRAM)**
  - Asymmetric performance / power consumption
  - Vertical integration → DRAM as cache
  - Horizontal integration → data placement

- **As a storage system (PCM & ReRAM)**
  - Vertical integration → example: hybrid disks
  - Horizontal integration → data placement controller ?, OS?, application ?

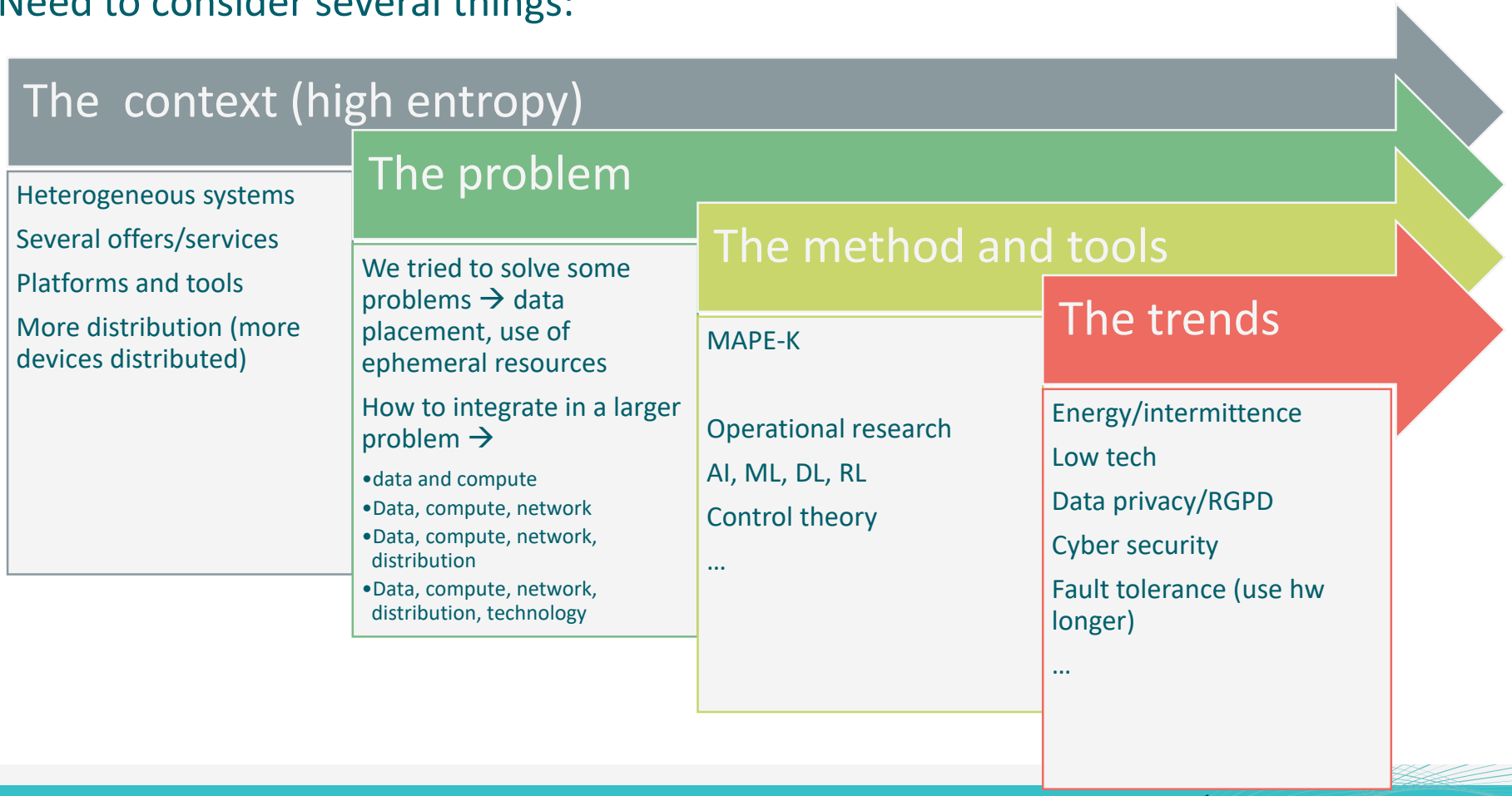
Integration in state-of-the-art work

		MRAM	eRAM	PCM	FeRAM
Cache (different levels)	Horizontal	[Oboril et al. 2015; Li et al. 2012; Syu et al. 2013; Li et al. 2014; Wu et al. 2009; Jadidi et al. 2011; Li et al. 2011; Wang et al. 2014 ; Komalan et al. 2014; Cheng et al. 2016]	[Wang et al. 2014a; Komalan et al. 2013; Mittal and Vetter 2015]	[Wu et al. 2009; Joo et al. 2010]	
	Vertical	[Oboril et al. 2015; Senni et al. 2014; Sun et al. 2009; Wu et al. 2009; Samavatian et al. 2014; Smullen et al. 2011; Jog et al. 2012; Zhou et al. 2009b; Goswami et al. 2013; Yazdanshenas et al. 2014; Ahn et al. 2012; Rasquinha et al. 2010; Park et al. 2012; Chen et al. 2013; Kwon et al. 2014; Jokar et al. 2016; Senni et al. 2015; Cheng et al. 2016]	[Dong et al. 2013; Wang et al. 2013; Jokar et al. 2016]	[Wu et al. 2009]	
	Replacement	[Oboril et al. 2015; Smullen et al. 2011; Sun et al. 2011; Guo et al. 2010; Goswami et al. 2013; Wang et al. 2015]	[Dong et al. 2013]		
Main Memory	Horizontal	[Yang et al. 2013; Suresh et al. 2014; Wei et al. 2015]	[Hassan et al. 2015; Wei et al. 2015]	[Dhiman et al. 2009; Park et al. 2010; Bock et al. 2011; Suresh et al. 2014; Zhou et al. 2009a; Sun et al. 2015; Wei et al. 2015; Lee et al. 2014; Salkhordeh and Asadi 2016; Wei et al. 2015; Kannan et al. 2016; Dulloor et al. 2016; Wu et al. 2016 ; Li et al. 2012 ; Oikawa 2014; Gao et al. 2015]	[Yoon et al. 2007; Suresh et al. 2014]
	Vertical	[Suresh et al. 2014]		[Qureshi and Srinivasan 2009; Suresh et al. 2014; Awad et al. 2016; Wu et al. 2016]	[Suresh et al. 2014; Jung et al. 2010]
	Replacement	[Kultursay et al. 2013; Wang et al. 2014; Jin et al. 2014]	[Xu et al. 2013 ; Xu et al. 2015]	[Lee et al. 2009; Chen et al. 2012; Park et al. 2015]	[Kaek et al. 2013]
Storage	Horizontal	[Lee et al. 2014]	[Tanakamaru et al. 2014; Sun et al. 2014; Fujii et al. 2012]	[Sun et al. 2010; Caulfield et al. 2010; Park et al. 2010]	[Yoon et al. 2008]
	Vertical	[Kang et al. 2015]		[Liu et al. 2011 ; Kang et al. 2015]	
	Replacement	[Lee et al. 2014]	[Jung et al. 2013]	[Akel et al. 2011; Kim et al. 2014]	[Kaek et al. 2013]

Jalil Boukhobza, Stéphane Rubini, Renhai Chen, Zili Shao, **Emerging NVM: A Survey on Architectural Integration and Research Challenges, ACM Transactions on Design Automation of Electronic Systems (TODAES),** 23(2), 14:1-14:32 (2018).

Conclusion

Need to consider several things:



Special thanks to …

https://www.ensta-bretagne.fr/boukhobza

Hamza Ouarnoughi

Djillali Boukhelef

Amina Chikhaoui

Jean-Emile Dartois

Med Islam Naas

Vincent Lannurien

Lydia Ait Oucheggou

Mohamed Handaoui

Sid Ahmed Yalles

Laurent Lemarchand

Olivier Barais

Kamel Boukhalfa

Philippe Raipin

Stéphane Rubini

Frank Singhoff

Yassine Hadjadj Aoul