9° Foro de Cloud Computing:
Arquitectura Cloud

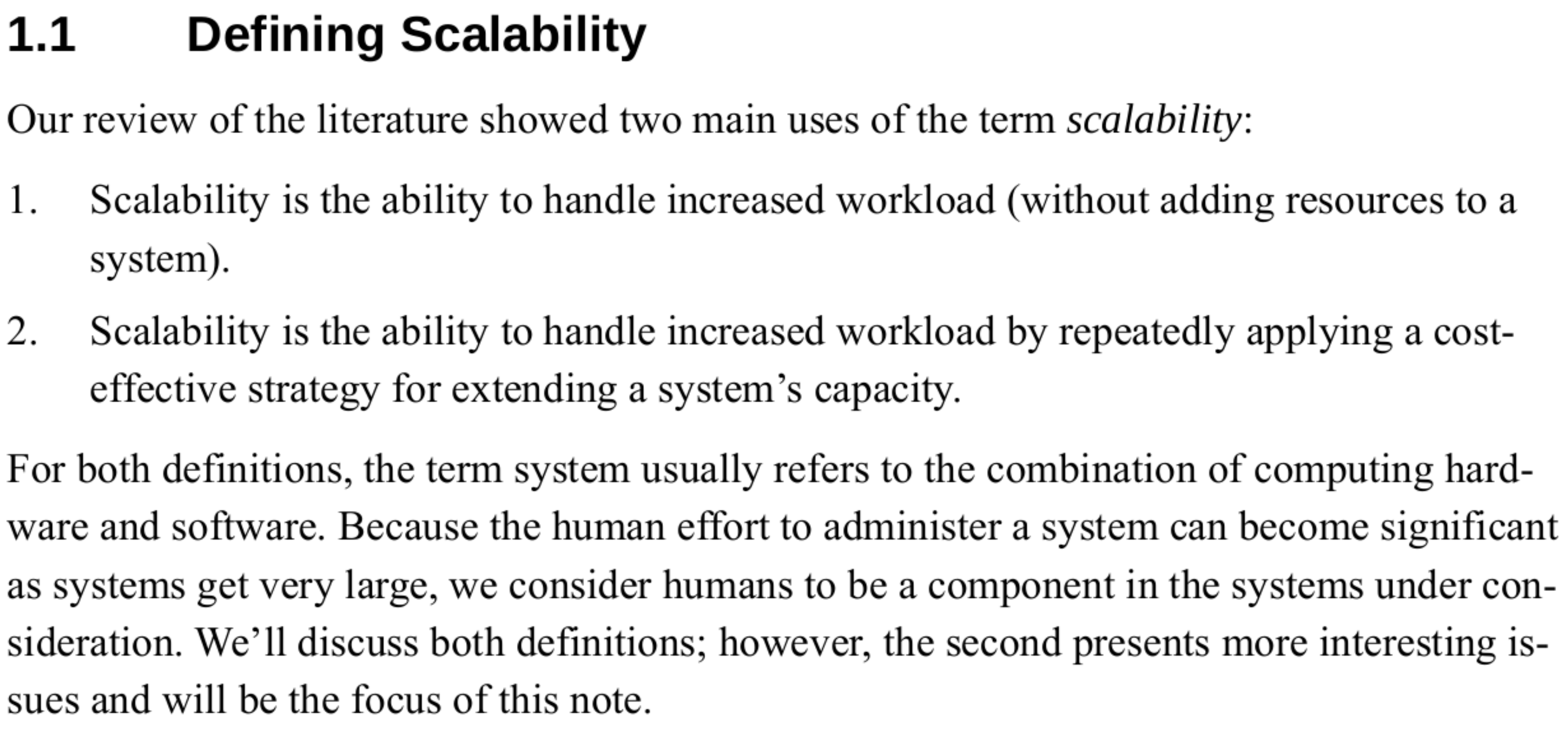
FOROS ISIS

Elasticidad: Grial de la Nube

Universidad de los Andes
Facultad de Ingeniería

Departamento de Ingeniería
de Sistemas y Computación

Capacidad de escalar

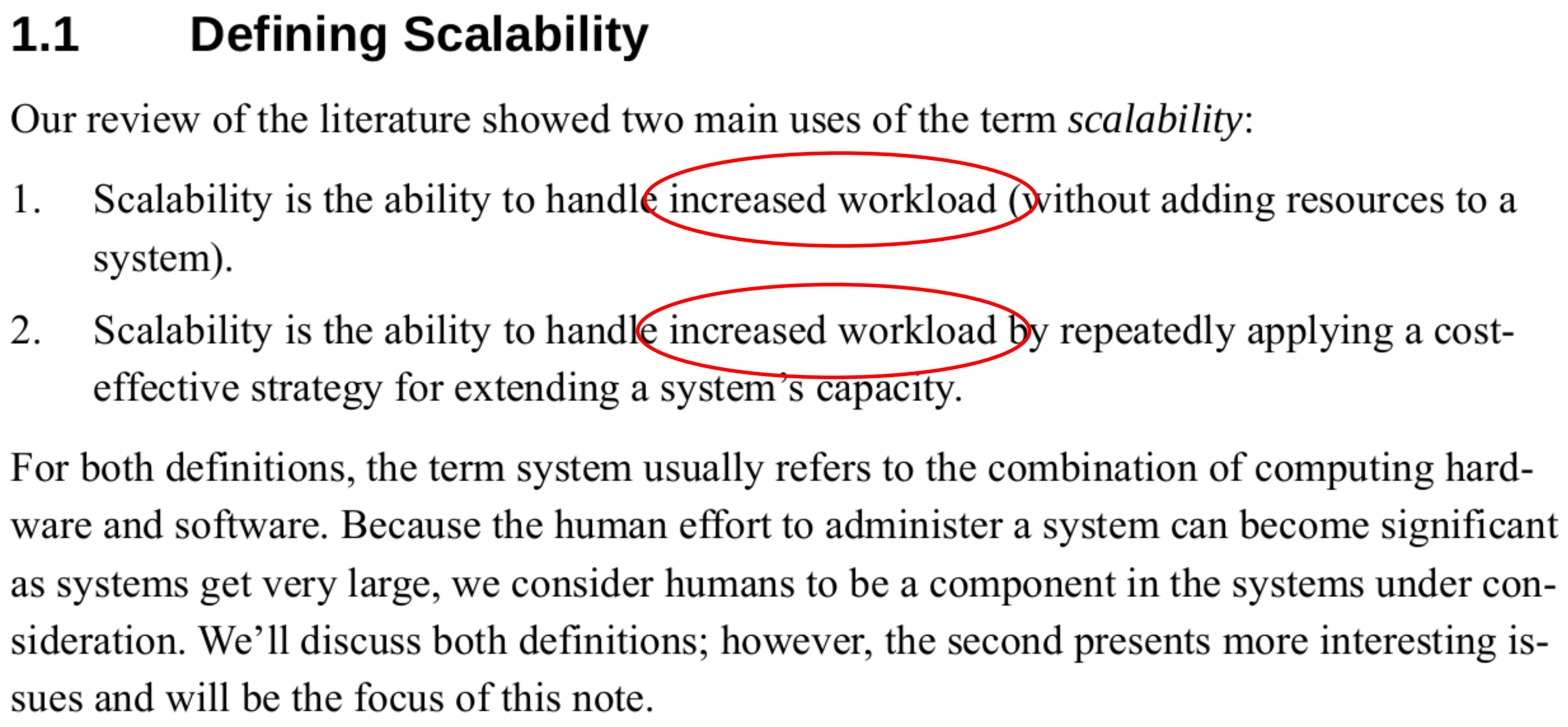


1.1 Defining Scalability

Our review of the literature showed two main uses of the term *scalability*:

1. Scalability is the ability to handle increased workload (without adding resources to a system).
2. Scalability is the ability to handle increased workload by repeatedly applying a cost-effective strategy for extending a system's capacity.

For both definitions, the term system usually refers to the combination of computing hardware and software. Because the human effort to administer a system can become significant as systems get very large, we consider humans to be a component in the systems under consideration. We'll discuss both definitions; however, the second presents more interesting issues and will be the focus of this note.



1.1 Defining Scalability

Our review of the literature showed two main uses of the term *scalability*:

1. Scalability is the ability to handle increased workload (without adding resources to a system).
2. Scalability is the ability to handle increased workload by repeatedly applying a cost-effective strategy for extending a system's capacity.

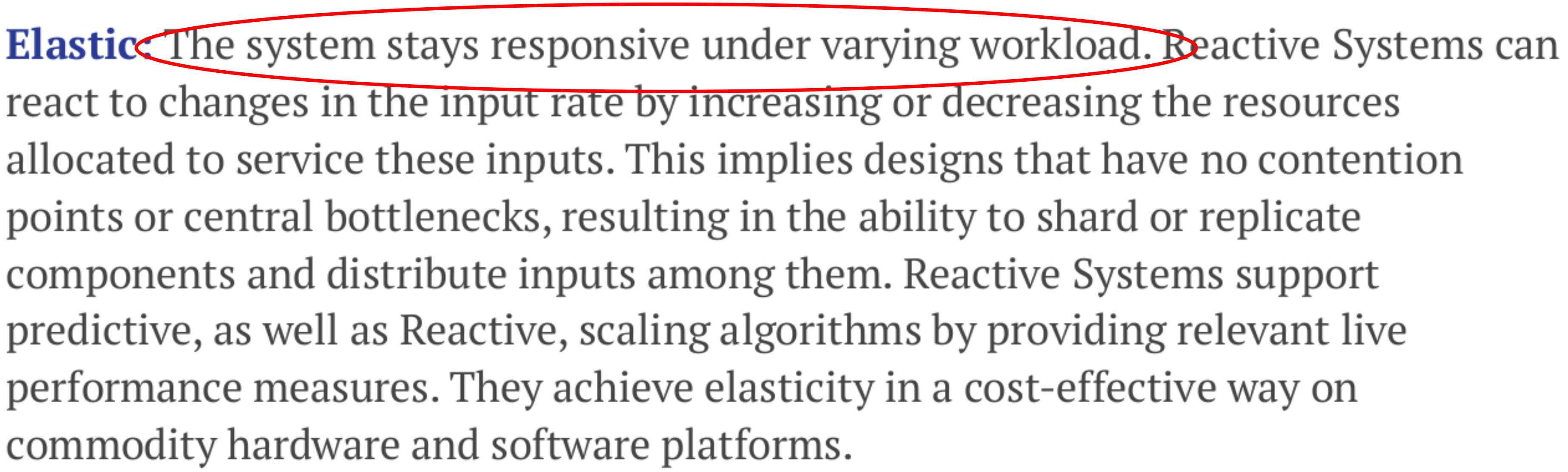
For both definitions, the term system usually refers to the combination of computing hardware and software. Because the human effort to administer a system can become significant as systems get very large, we consider humans to be a component in the systems under consideration. We'll discuss both definitions; however, the second presents more interesting issues and will be the focus of this note.

¿más es más?

¿Y el menos?

Elasticidad

Elastic: The system stays responsive under varying workload. Reactive Systems can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs. This implies designs that have no contention points or central bottlenecks, resulting in the ability to shard or replicate components and distribute inputs among them. Reactive Systems support predictive, as well as Reactive, scaling algorithms by providing relevant live performance measures. They achieve elasticity in a cost-effective way on commodity hardware and software platforms.



Elastic: The system stays responsive under varying workload. Reactive Systems can react to changes in the input rate by increasing or decreasing the resources allocated to service these inputs. This implies designs that have no contention points or central bottlenecks, resulting in the ability to shard or replicate components and distribute inputs among them. Reactive Systems support predictive, as well as Reactive, scaling algorithms by providing relevant live performance measures. They achieve elasticity in a cost-effective way on commodity hardware and software platforms.

Elasticidad es **un medio** para alcanzar un fin: **desempeño** y/o **capacidad**

La promesa.

Tenemos **elasticidad** ilimitada.

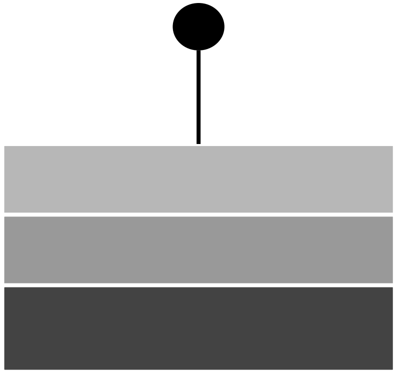
Paga por lo que **consume**.

El problema.

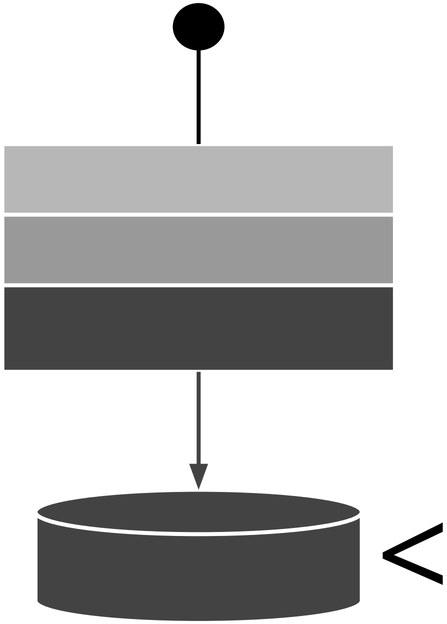
Tipo de aplicaciones

Sin estado.
Con estado.

Sin estado.



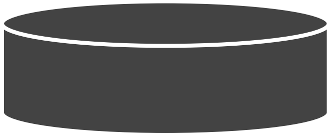
Funciones *puras*

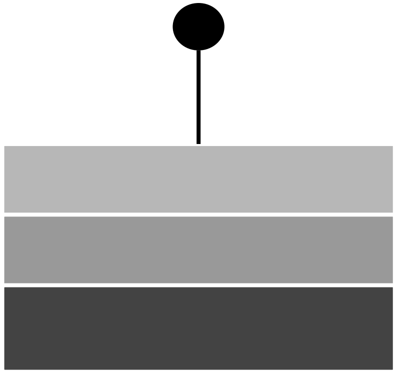


Func. no tan *puras*

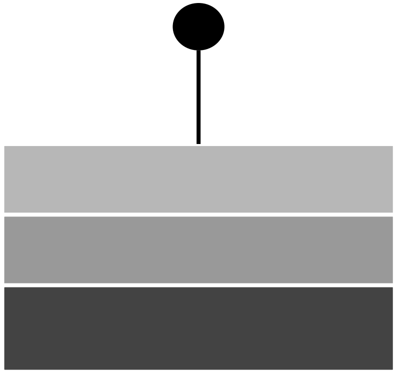
< ¡Estado!

Con estado.

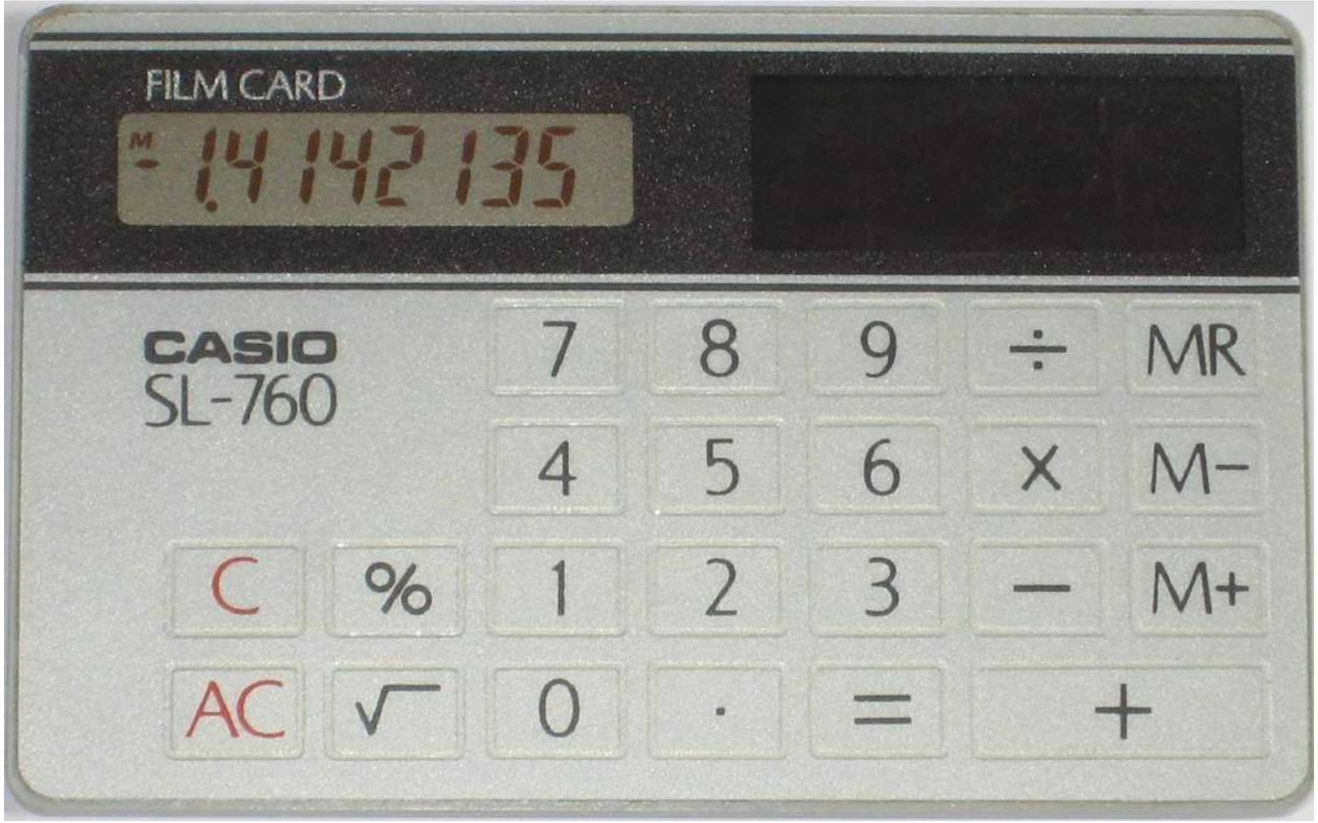
 < ¡Estado!



< ¡Estado!

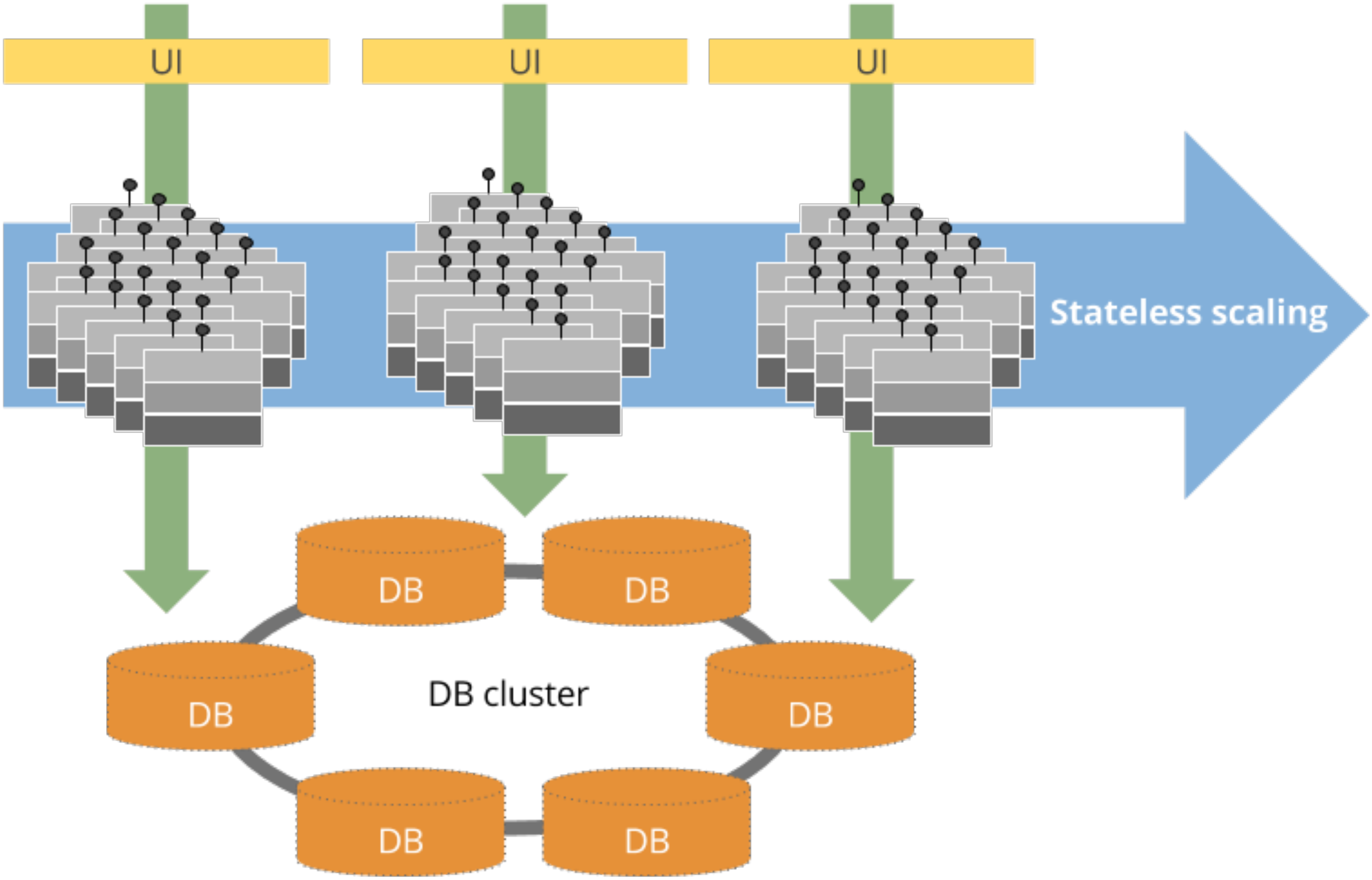


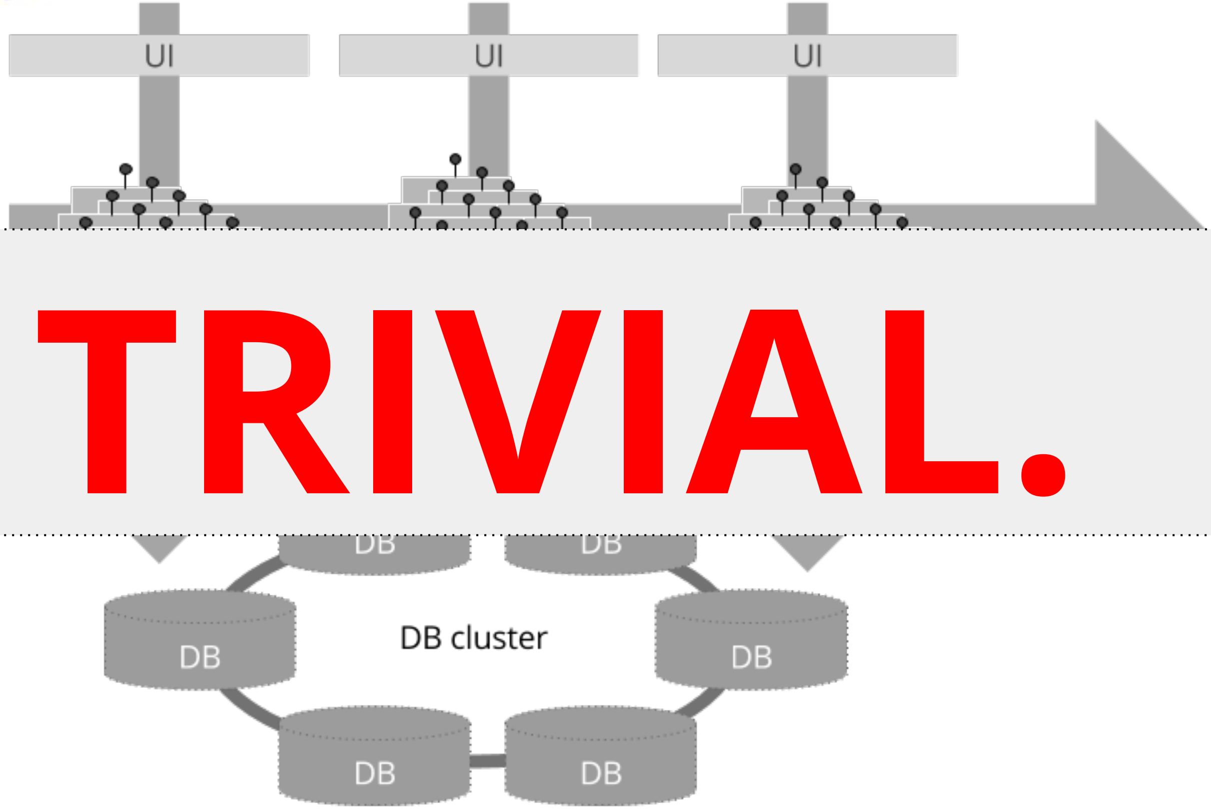
Cache de datos


FILM CARD
M
-1.4142135
CASIO
SL-760
7
8
9
÷
MR
4
5
6
×
M−
C
%
1
2
3
−
M+
AC
√
0
·
=
+

El problema.

Escalar apps **sin estado.**


UI
UI
UI
Stateless scaling
DB
DB
DB
DB cluster
DB
DB
DB



TRIVIAL.

9° Foro de Cloud Computing: Arquitectura Cloud

Scalability! But at what COST?

Frank McSherry
Unaffiliated

Michael Isard
Microsoft Research

Derek G. Murray
Unaffiliated*

Abstract

We offer a new metric for big data platforms, COST, or the Configuration that Outperforms a Single Thread. The COST of a given platform for a given problem is the hardware configuration required before the platform outperforms a competent single-threaded implementation. COST weighs a system's scalability against the overheads introduced by the system, and indicates the actual performance gains of the system, without rewarding systems that bring substantial but parallelizable overheads.

We survey measurements of data-parallel systems re-

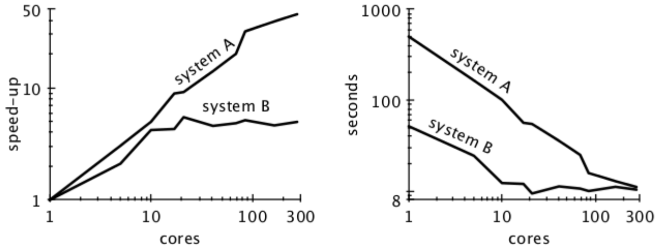
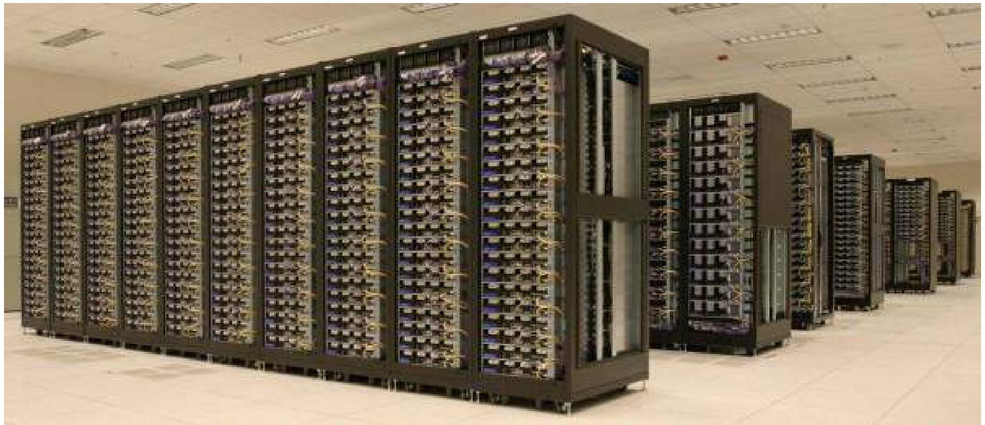


Figure 1: Scaling and performance measurements for a data-parallel algorithm, before (system A) and after (system B) a simple performance optimization. The unoptimized implementation "scales" far better,

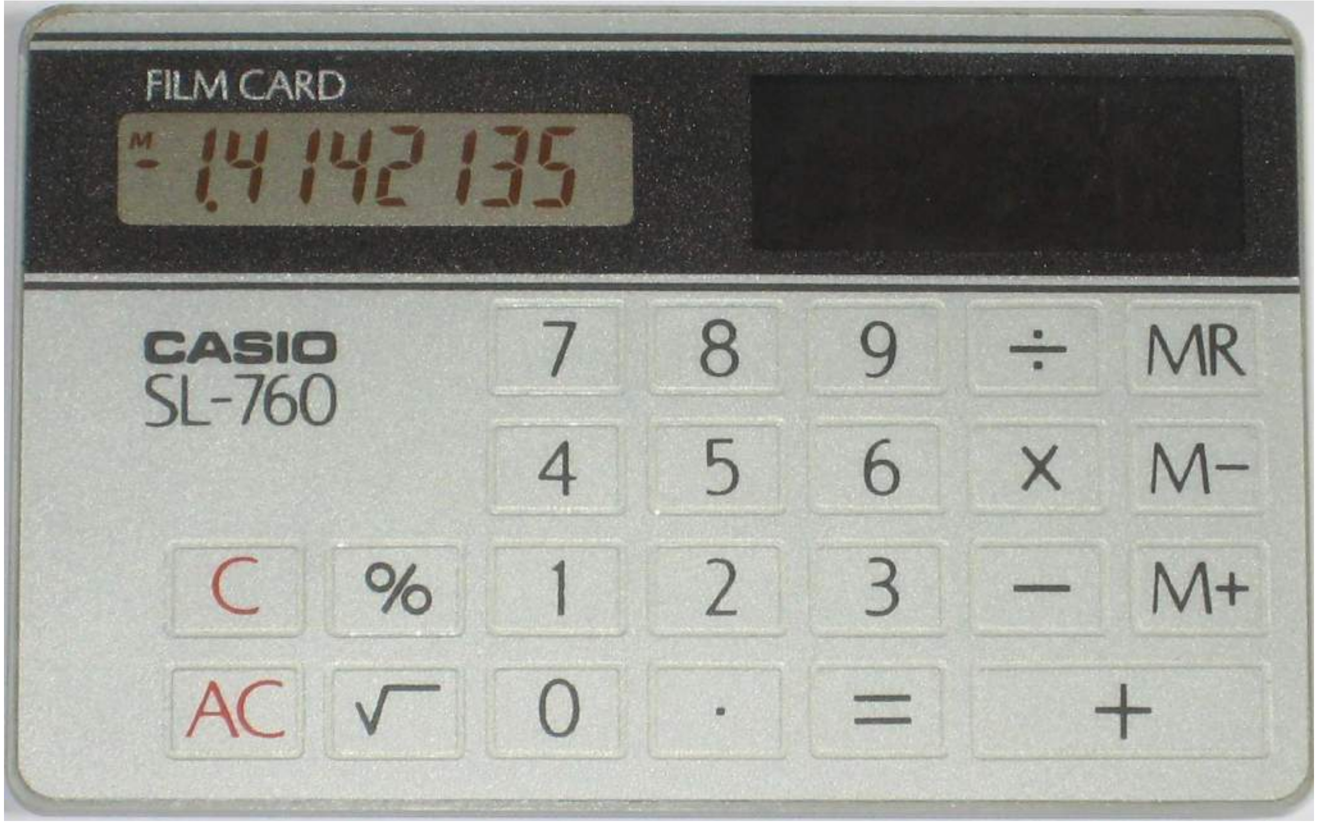
“**Rather** than making your computation **go faster**, the systems introduce substantial **overheads** which can require large compute clusters just to bring under control.”

>



El problema.

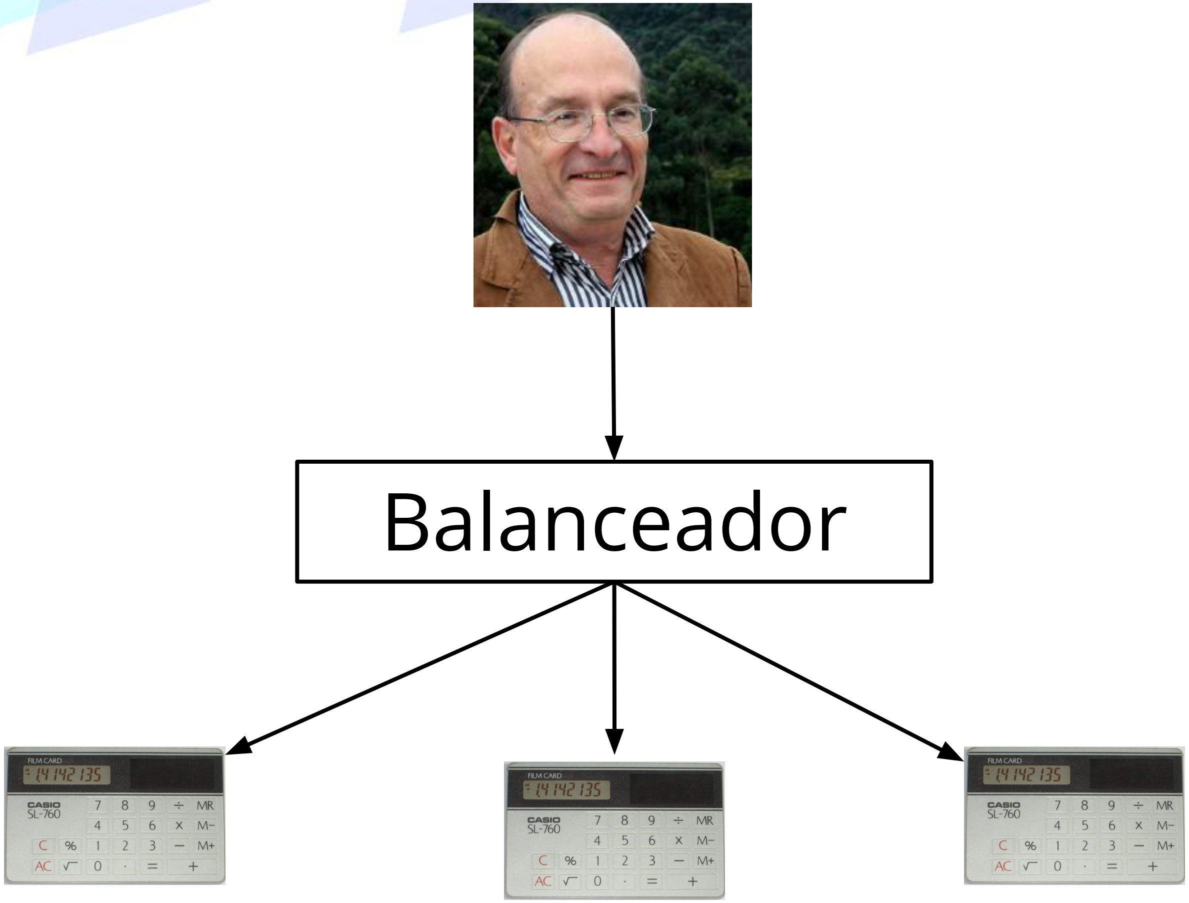
Escalar apps **con estado**.


FILM CARD
M
-1.4142135
CASIO
SL-760
7
8
9
÷
MR
4
5
6
×
M−
C
%
1
2
3
−
M+
AC
√
0
·
=
+

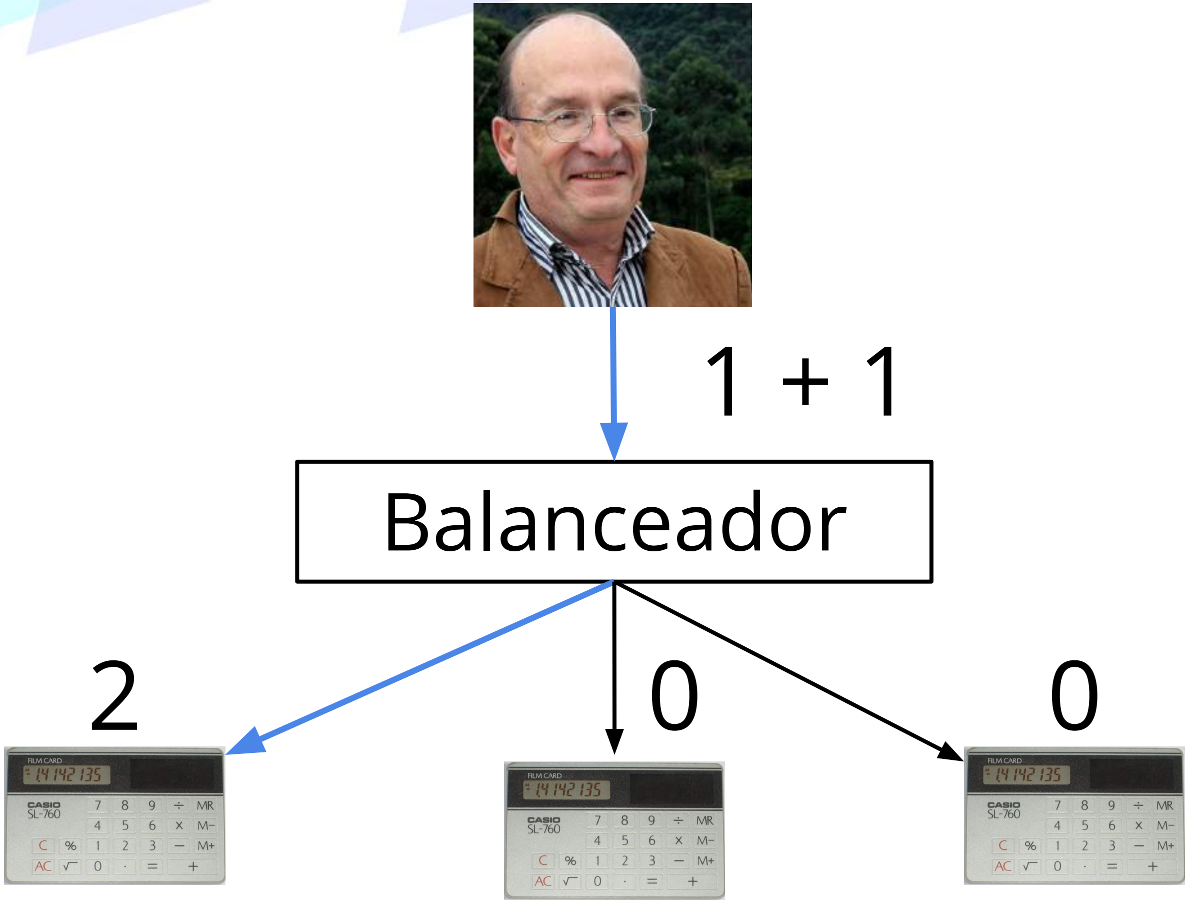
FILM CARD
1.4142135
CASIO
SL-760

FILM CARD
1.4142135
CASIO
SL-760

FILM CARD
1.4142135
CASIO
SL-760



Balanceador



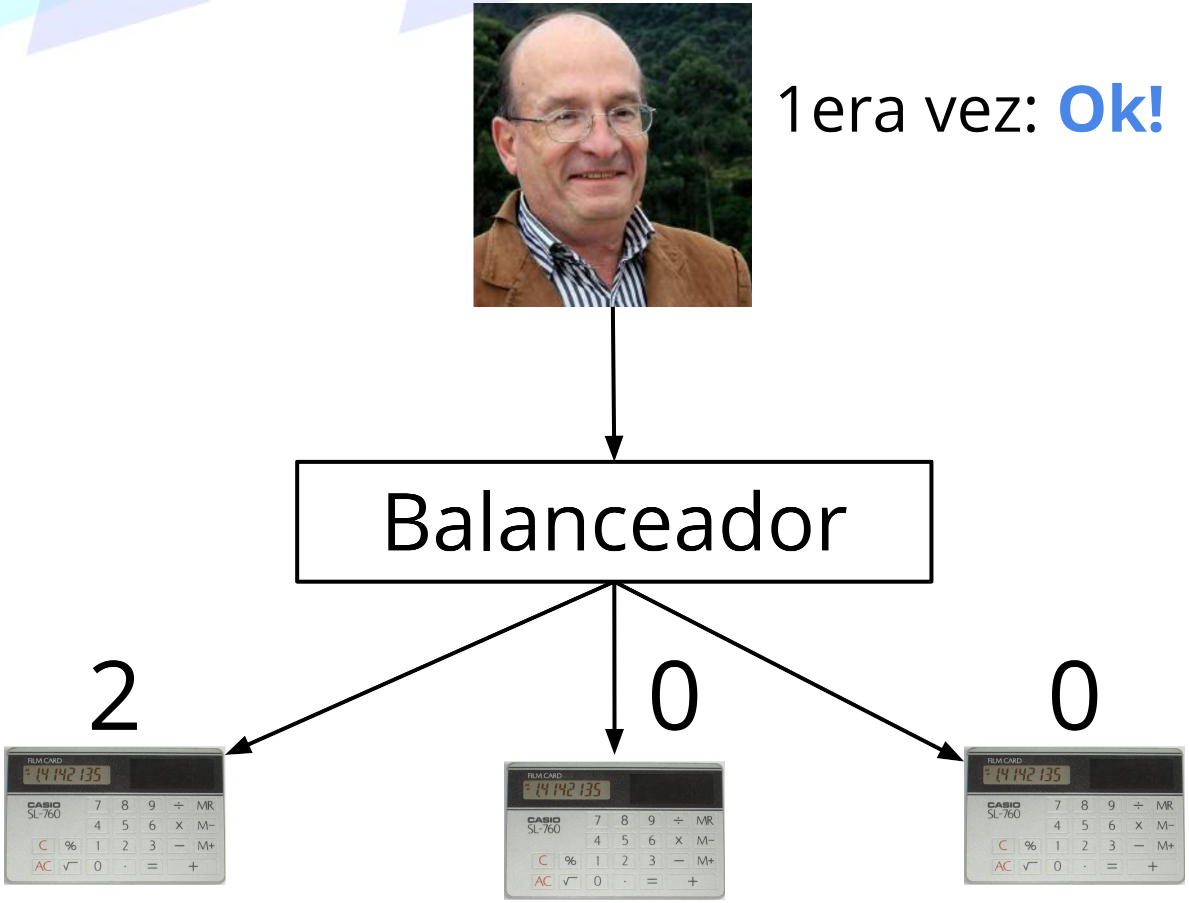
1 + 1

Balanceador

2

0

0

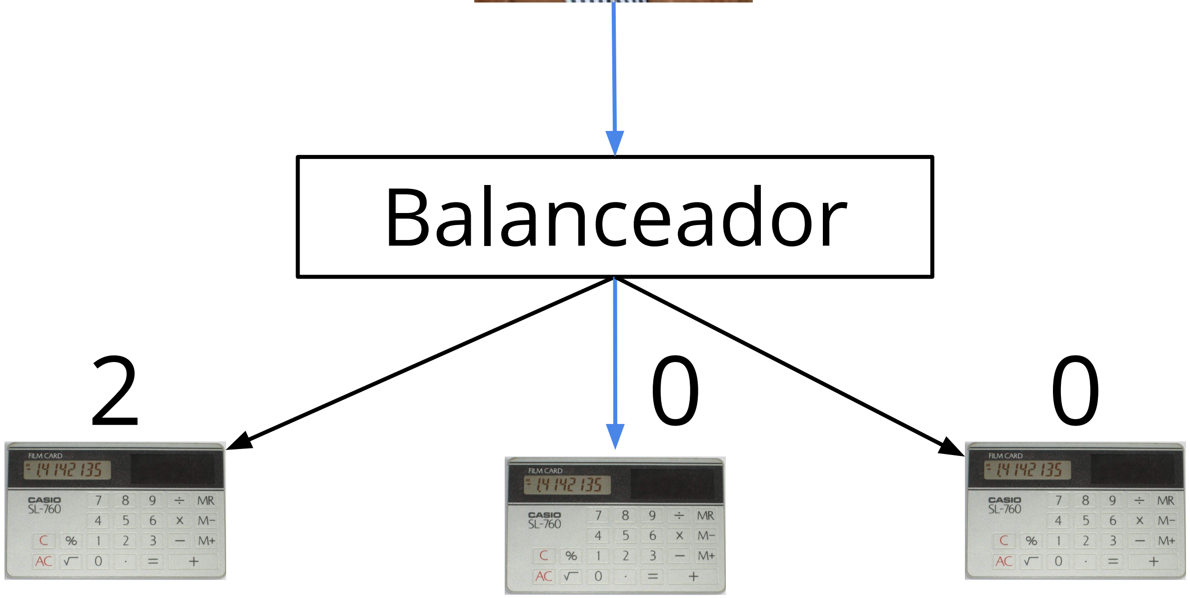


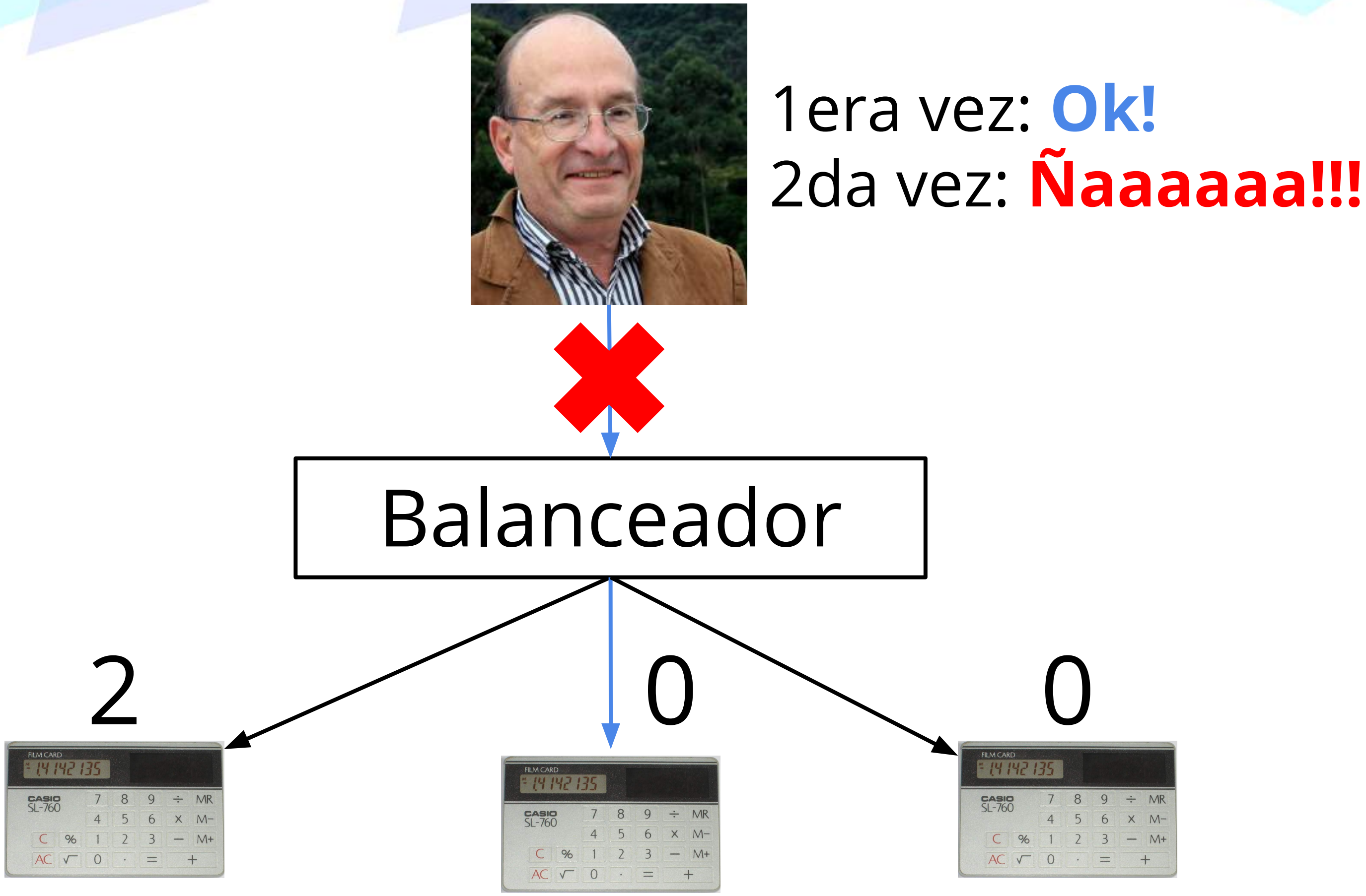
1era vez: **Ok!**

Balanceador

2 0 0

1era vez: **Ok!**
2da vez:





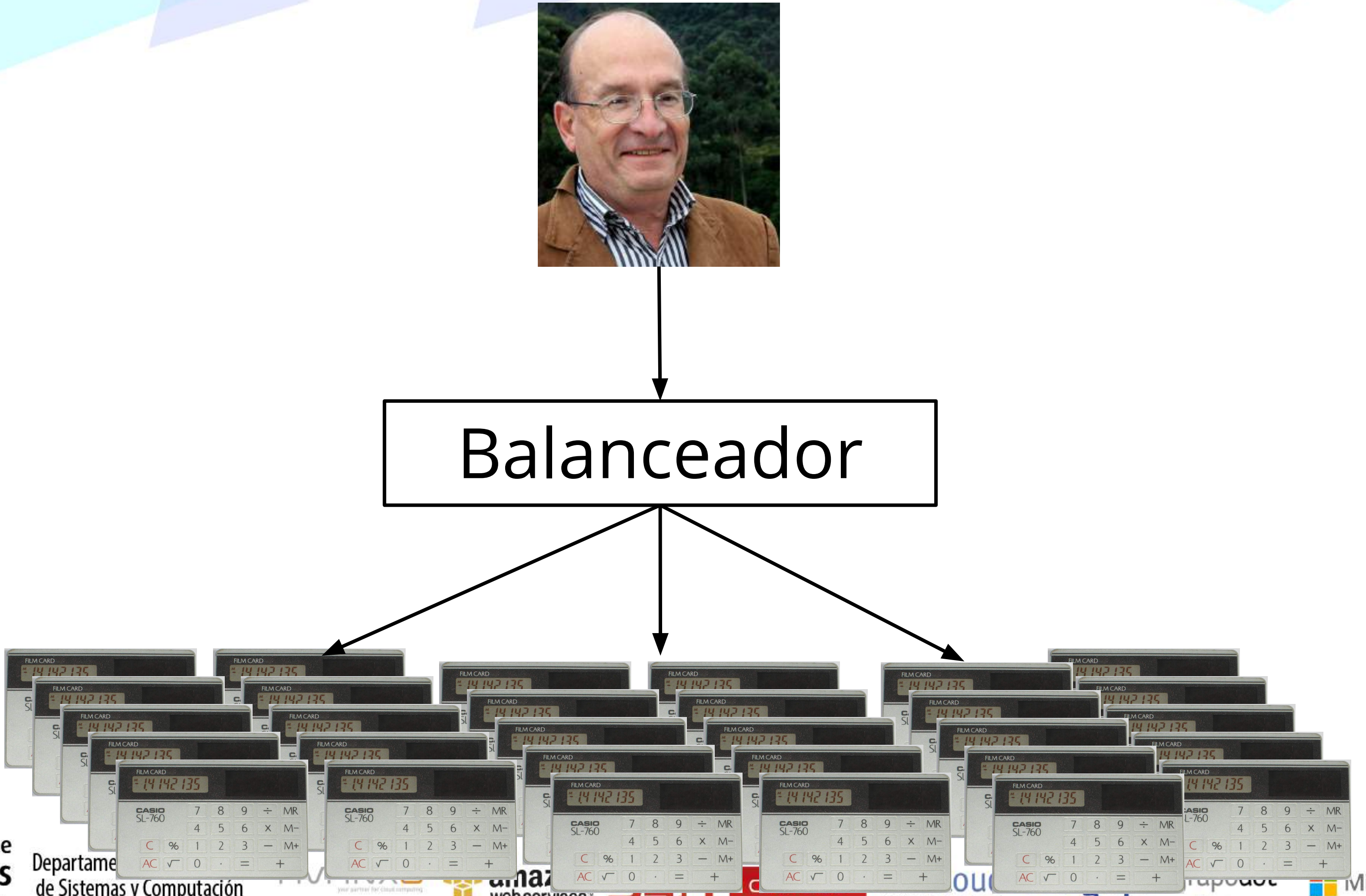
1era vez: **Ok!**
2da vez: **Ñaaaaaa!!!**

Balanceador

2

0

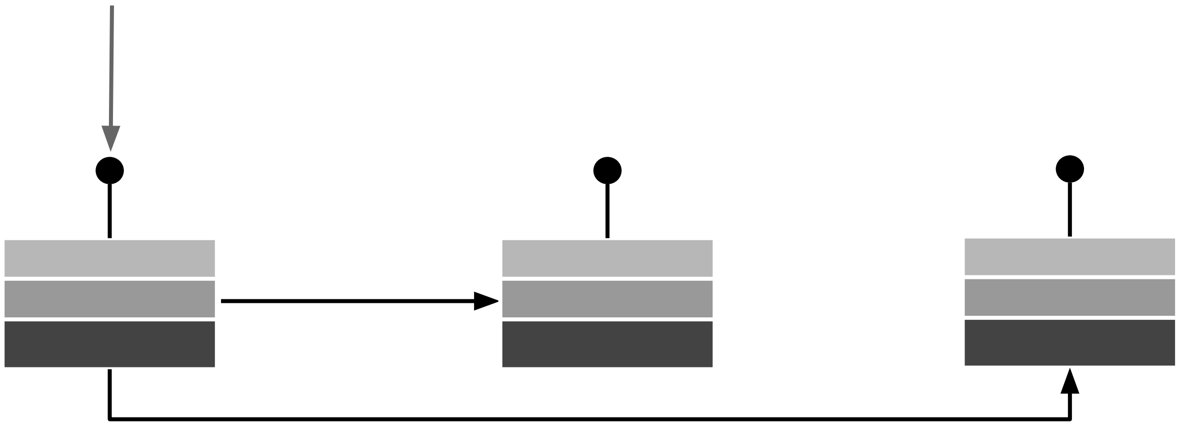
0

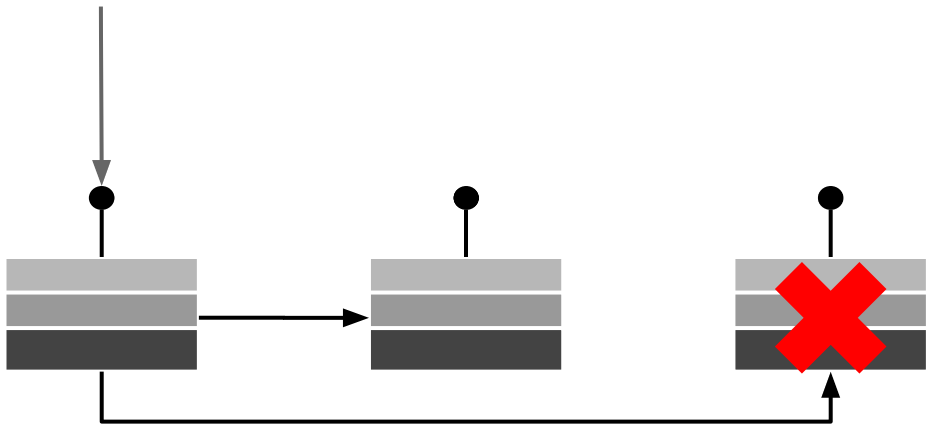


Balanceador

Nuestro camino a la **“solución”**.

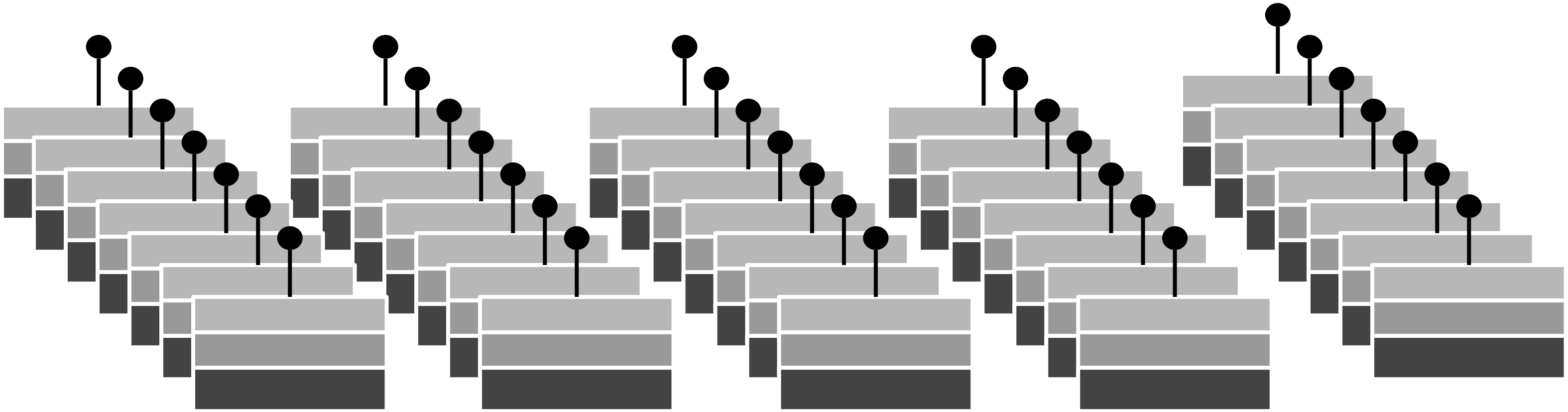
Intuición: **replicar** el estado para que todas las instancias *sepan*.





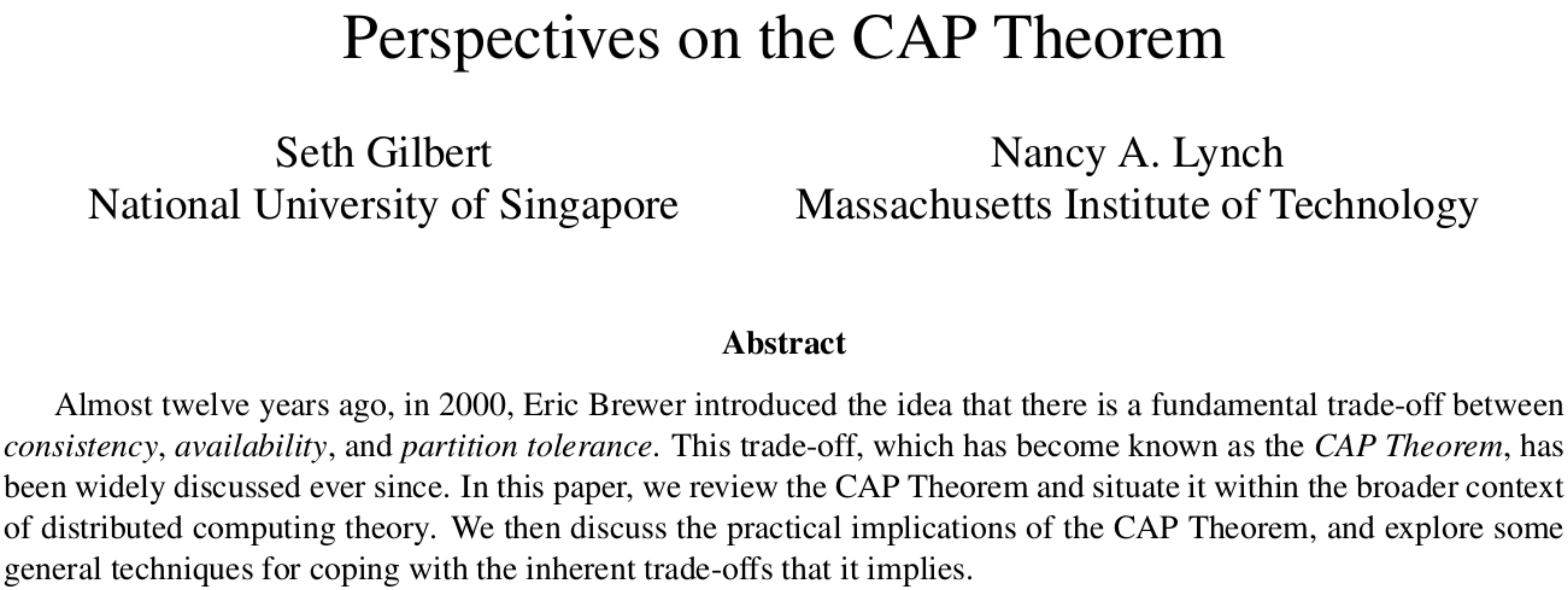
9° Foro de Cloud Computing: Arquitectura Cloud

FOROS ISIS



Alternativa #2

(Toca invocar poderes)



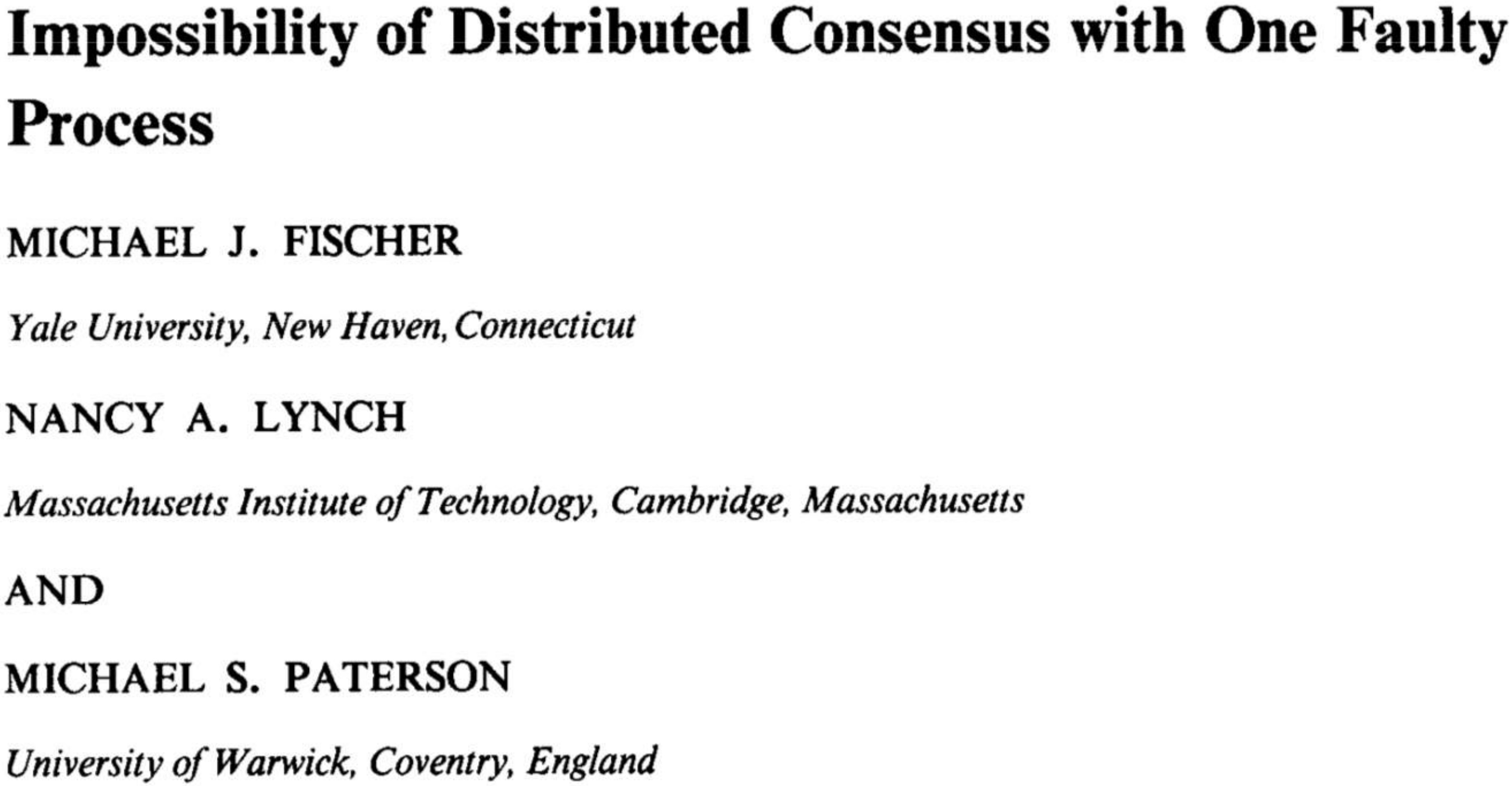
Perspectives on the CAP Theorem

Seth Gilbert
National University of Singapore

Nancy A. Lynch
Massachusetts Institute of Technology

Abstract

Almost twelve years ago, in 2000, Eric Brewer introduced the idea that there is a fundamental trade-off between *consistency*, *availability*, and *partition tolerance*. This trade-off, which has become known as the *CAP Theorem*, has been widely discussed ever since. In this paper, we review the CAP Theorem and situate it within the broader context of distributed computing theory. We then discuss the practical implications of the CAP Theorem, and explore some general techniques for coping with the inherent trade-offs that it implies.



Impossibility of Distributed Consensus with One Faulty Process

MICHAEL J. FISCHER

Yale University, New Haven, Connecticut

NANCY A. LYNCH

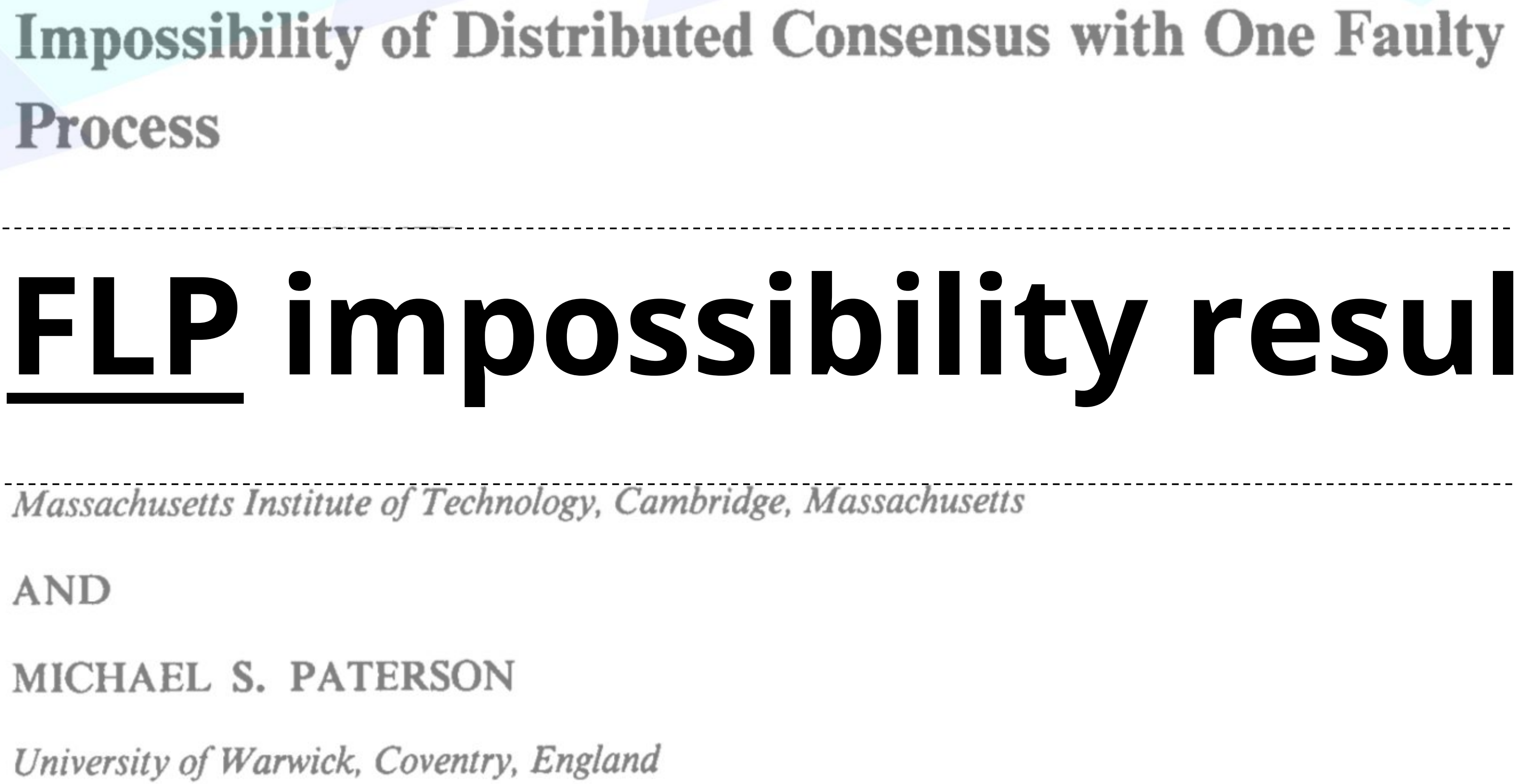
Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England

FLP impossibility result



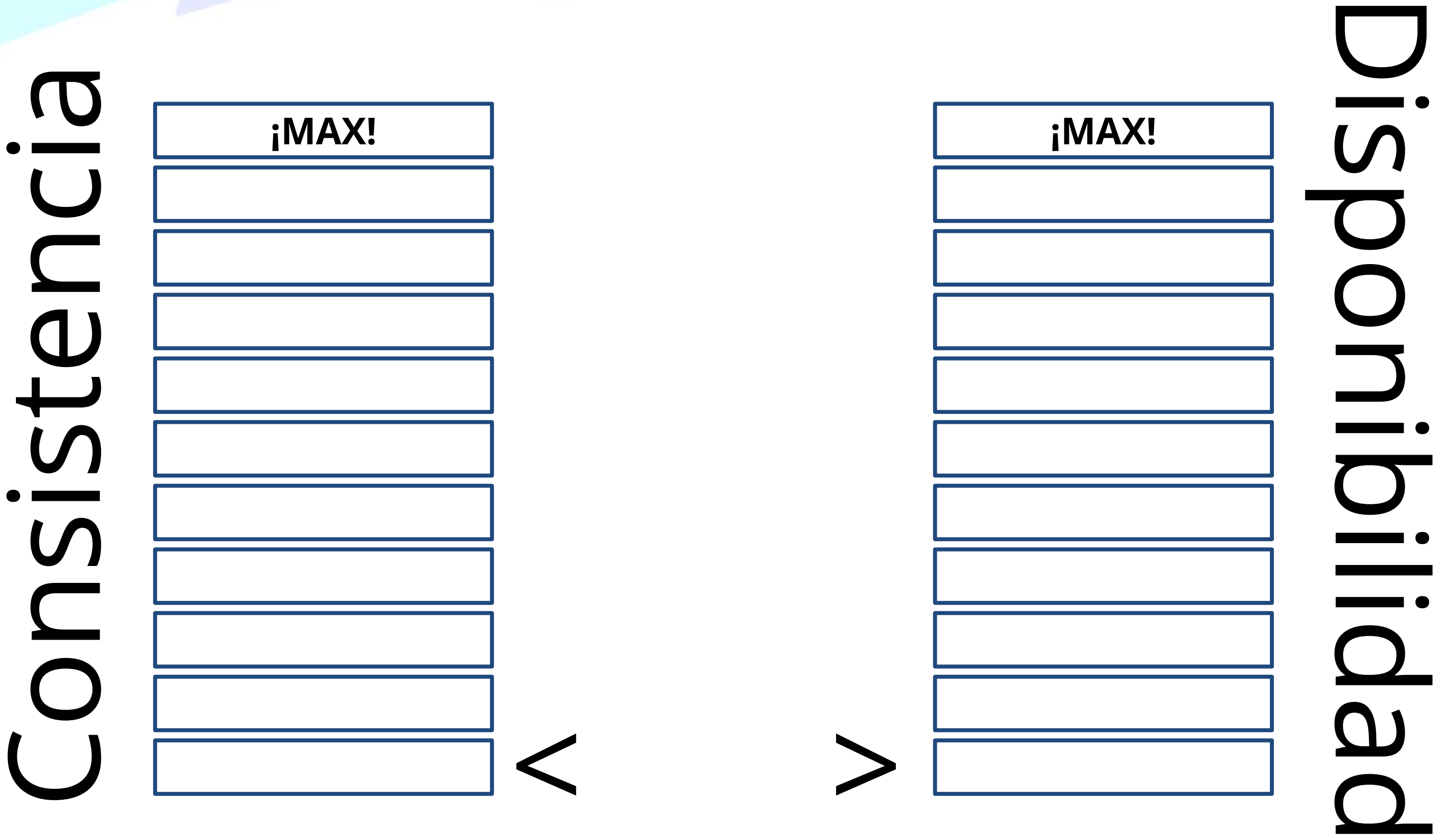
Impossibility of Distributed Consensus with One Faulty Process

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

MICHAEL S. PATERSON

University of Warwick, Coventry, England



Consistencia

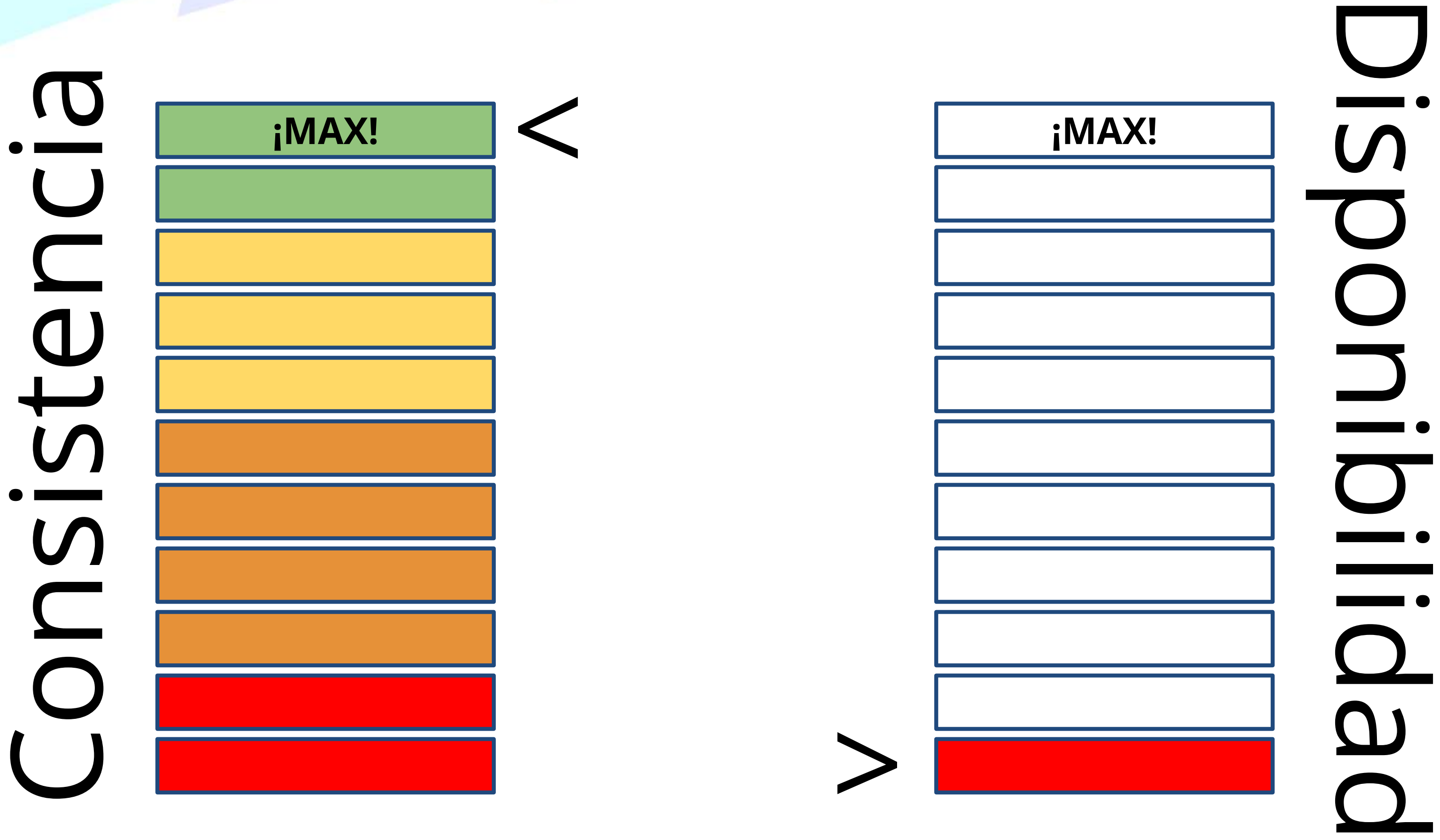
¡MAX!

<

>

¡MAX!

Disponibilidad



Consistencia

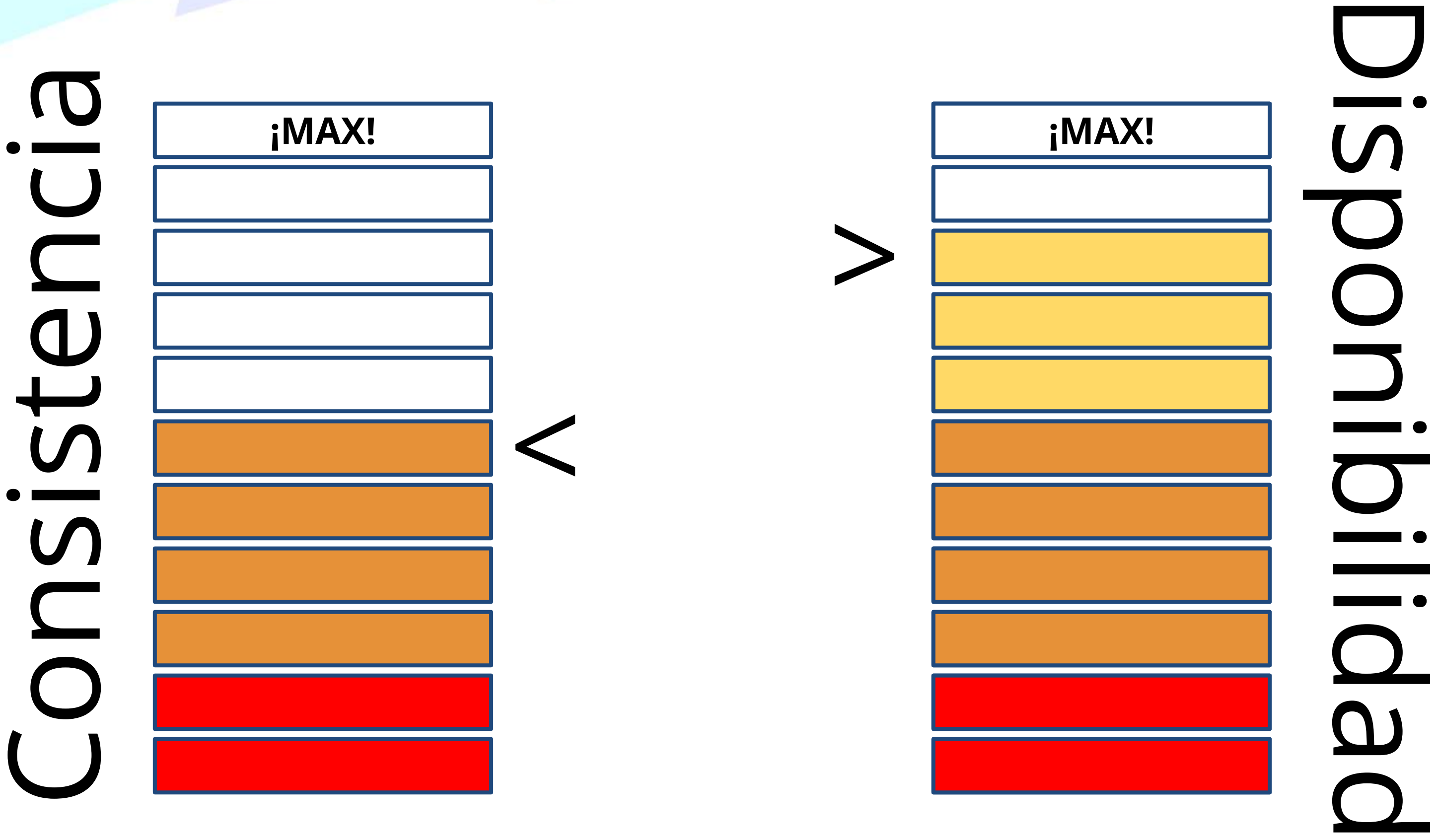
¡MAX!

<

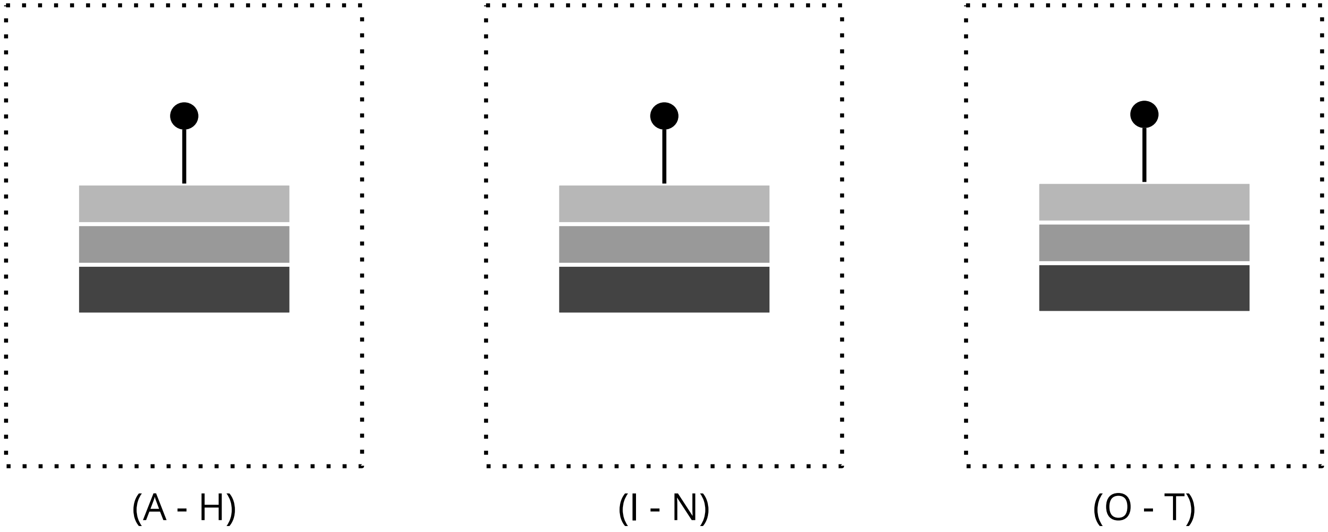
>

¡MAX!

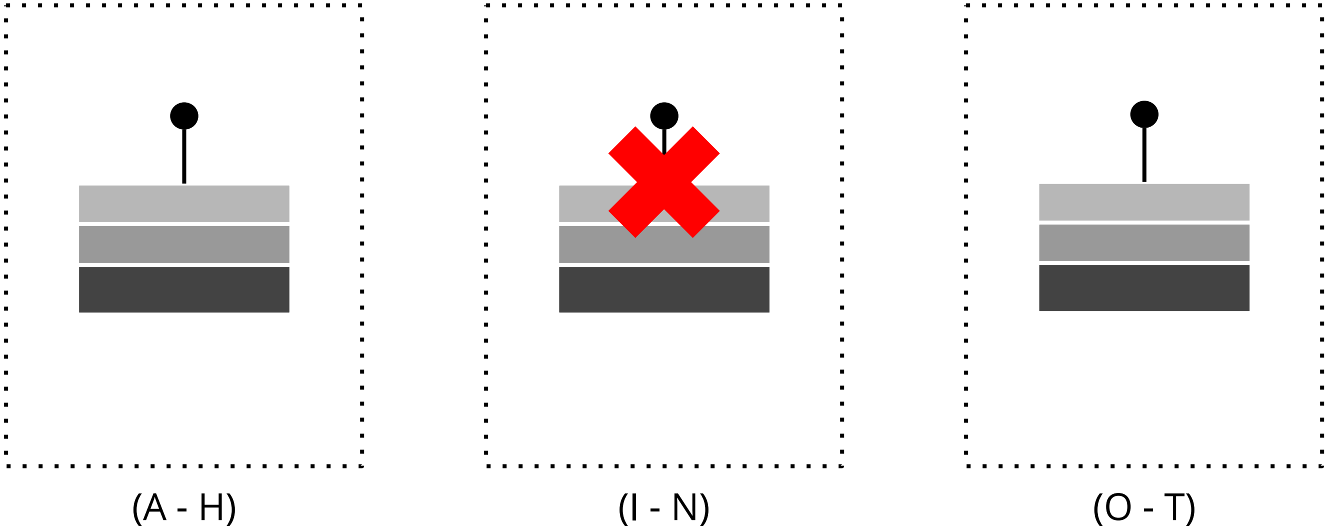
Disponibilidad


Consistencia
¡MAX!
<
>
¡MAX!
Disponibilidad

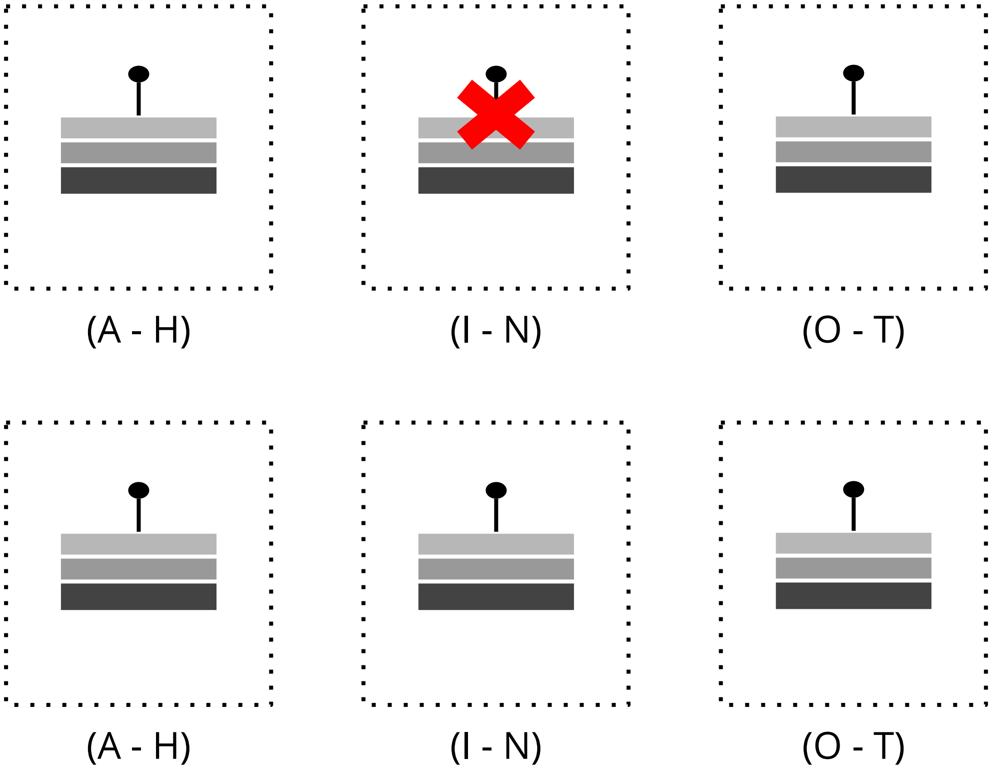
Particionar



Particionar

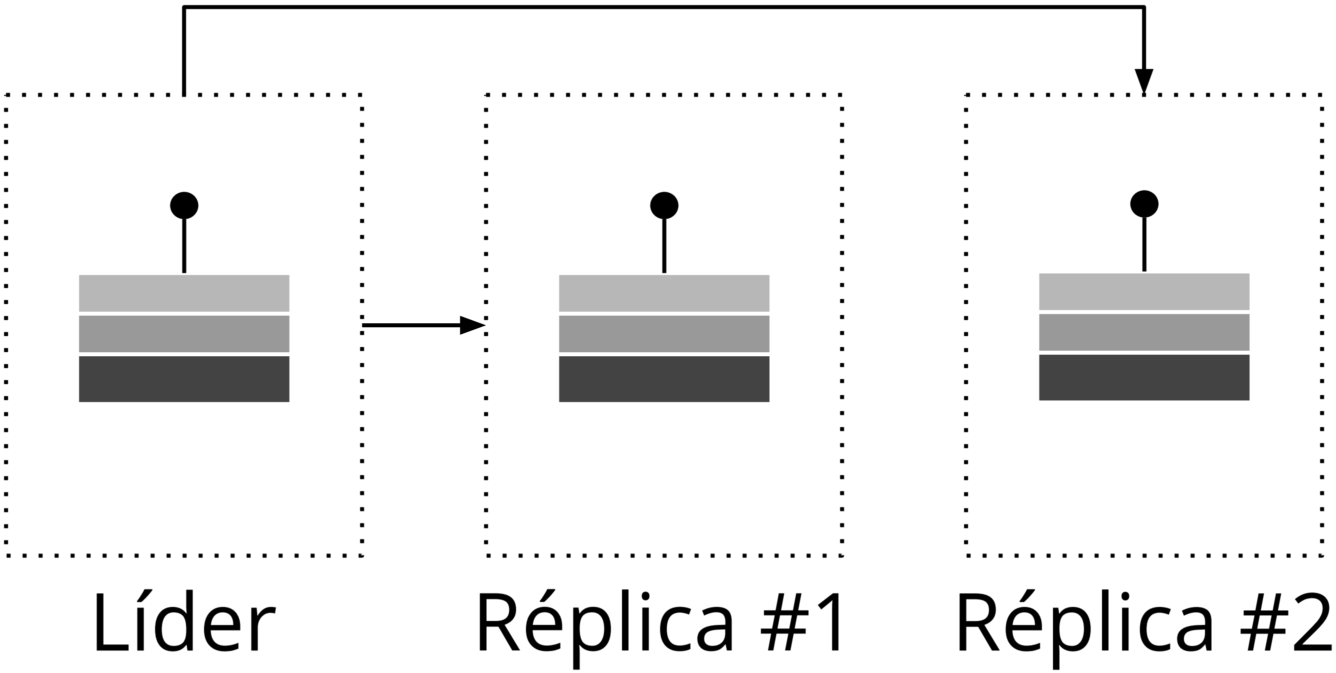


Particionar



Replicas

Quórum

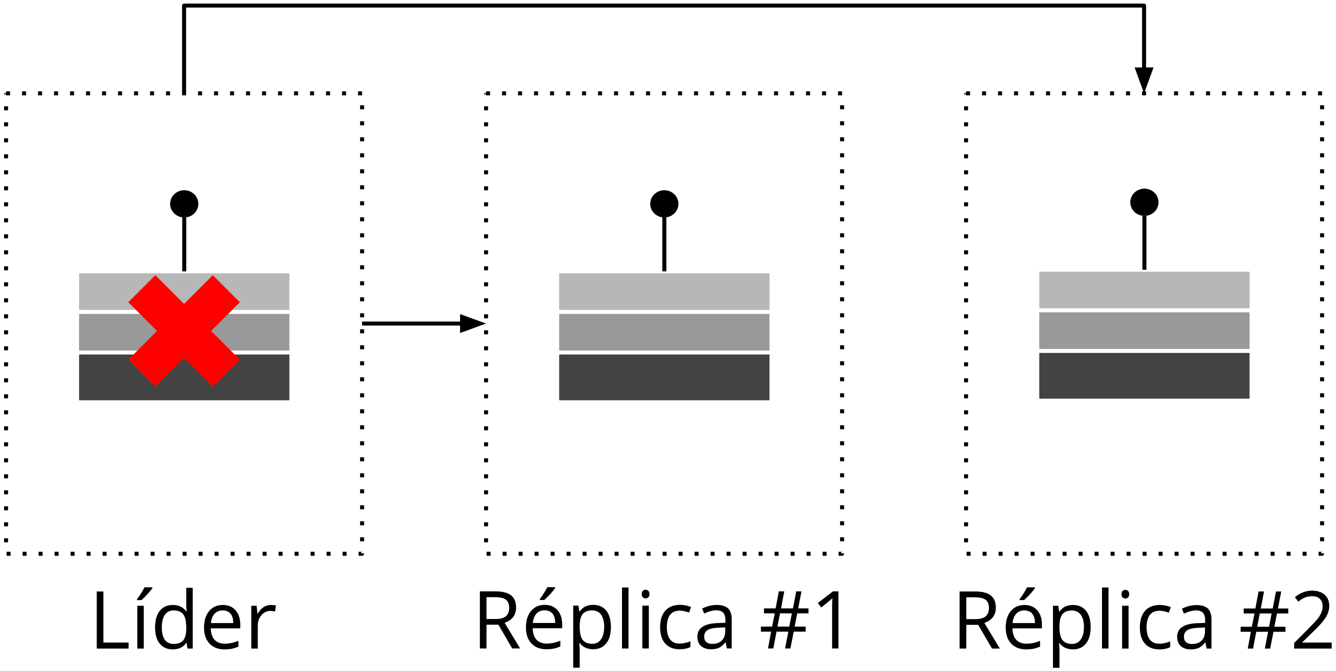


Líder

Réplica #1

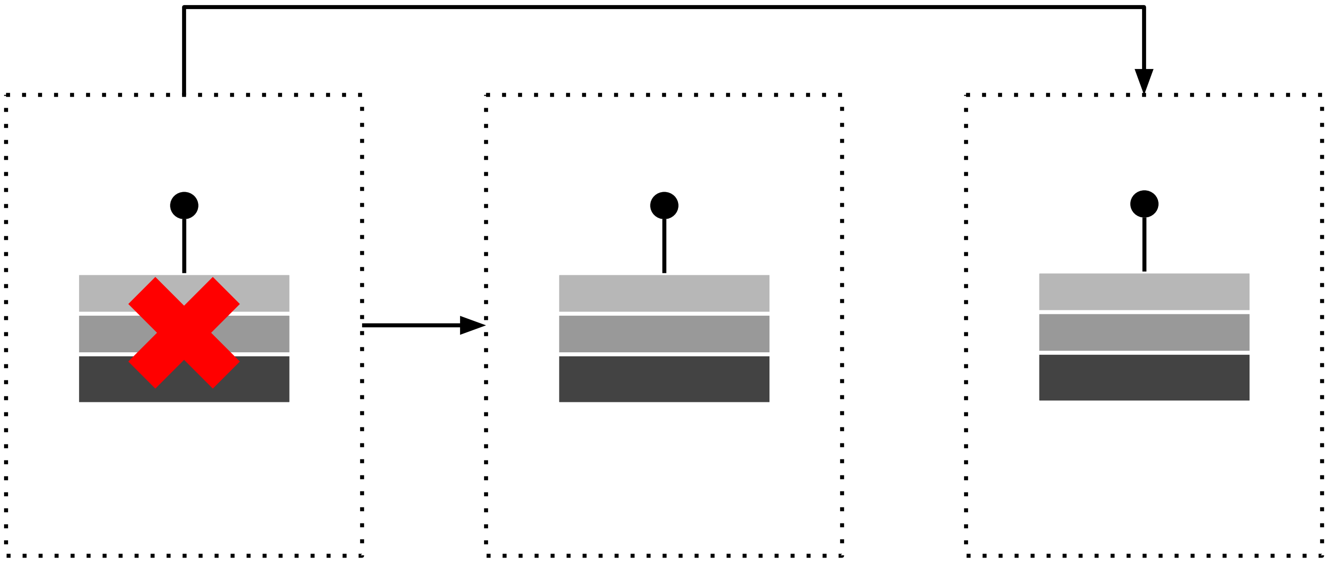
Réplica #2

Quórum



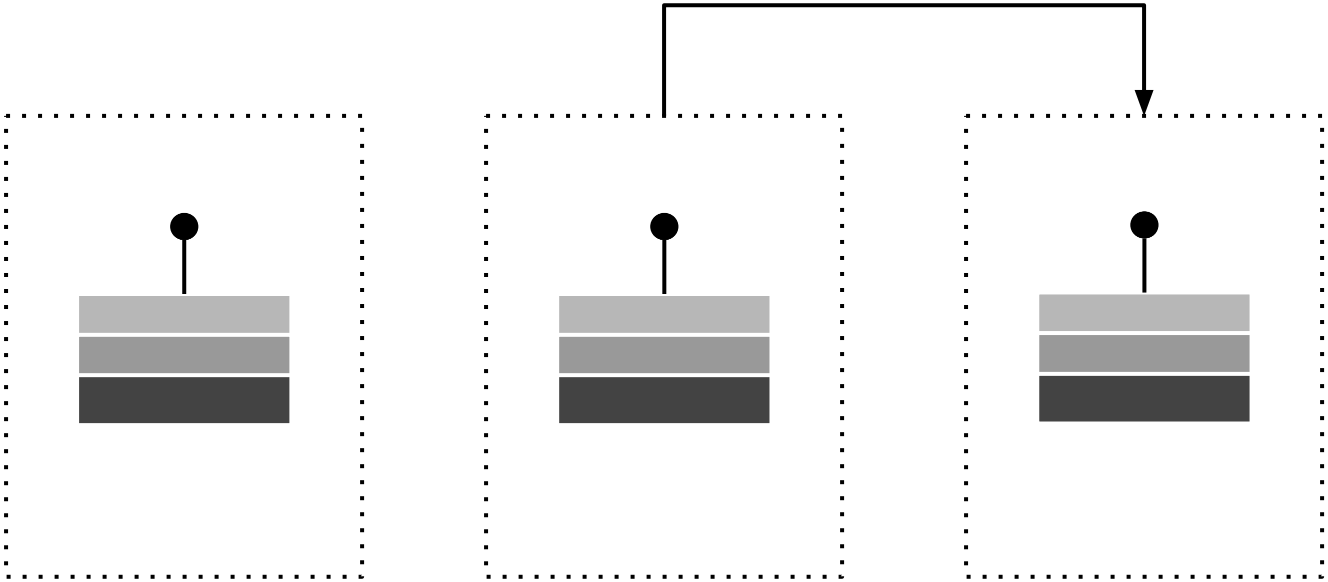
Cuando el nodo caído se recupera **¿qué?**

Quórum



Líder Réplica #1 Réplica #2

Quórum

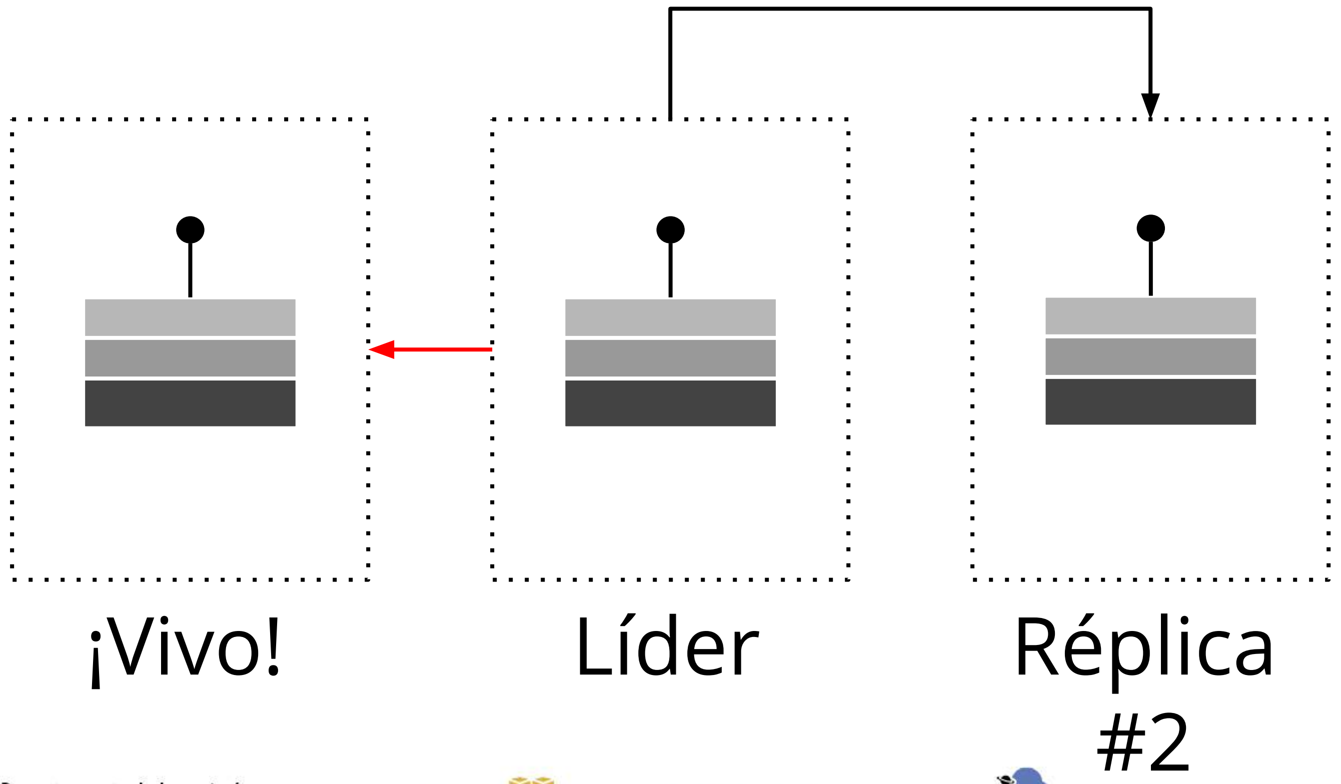


Murió :(

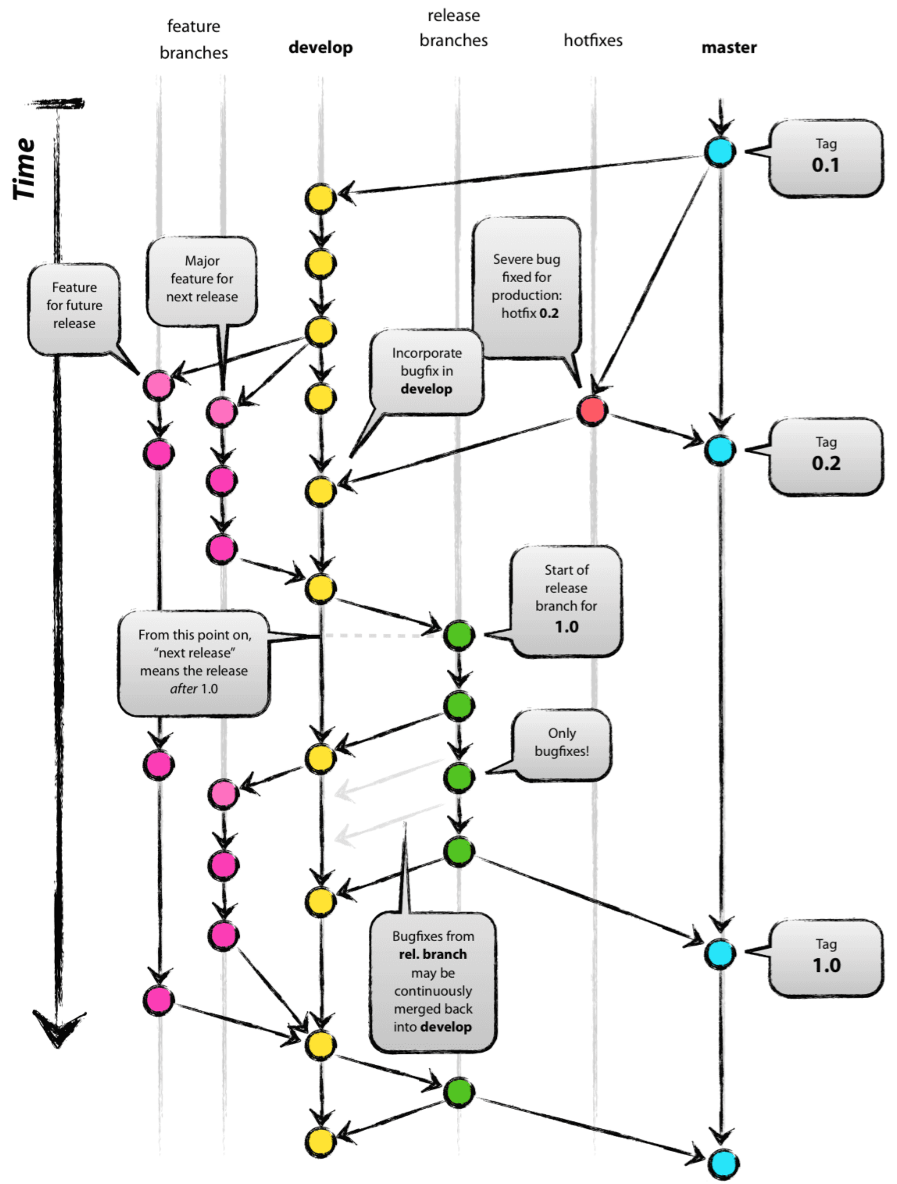
Líder

Réplica #2

Quórum

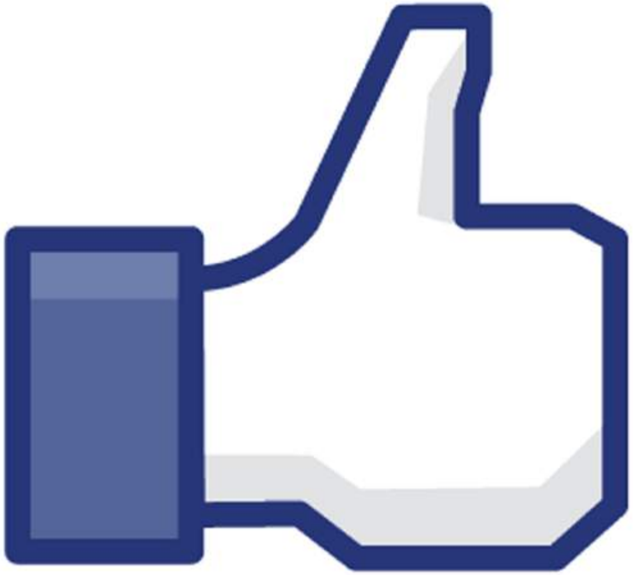


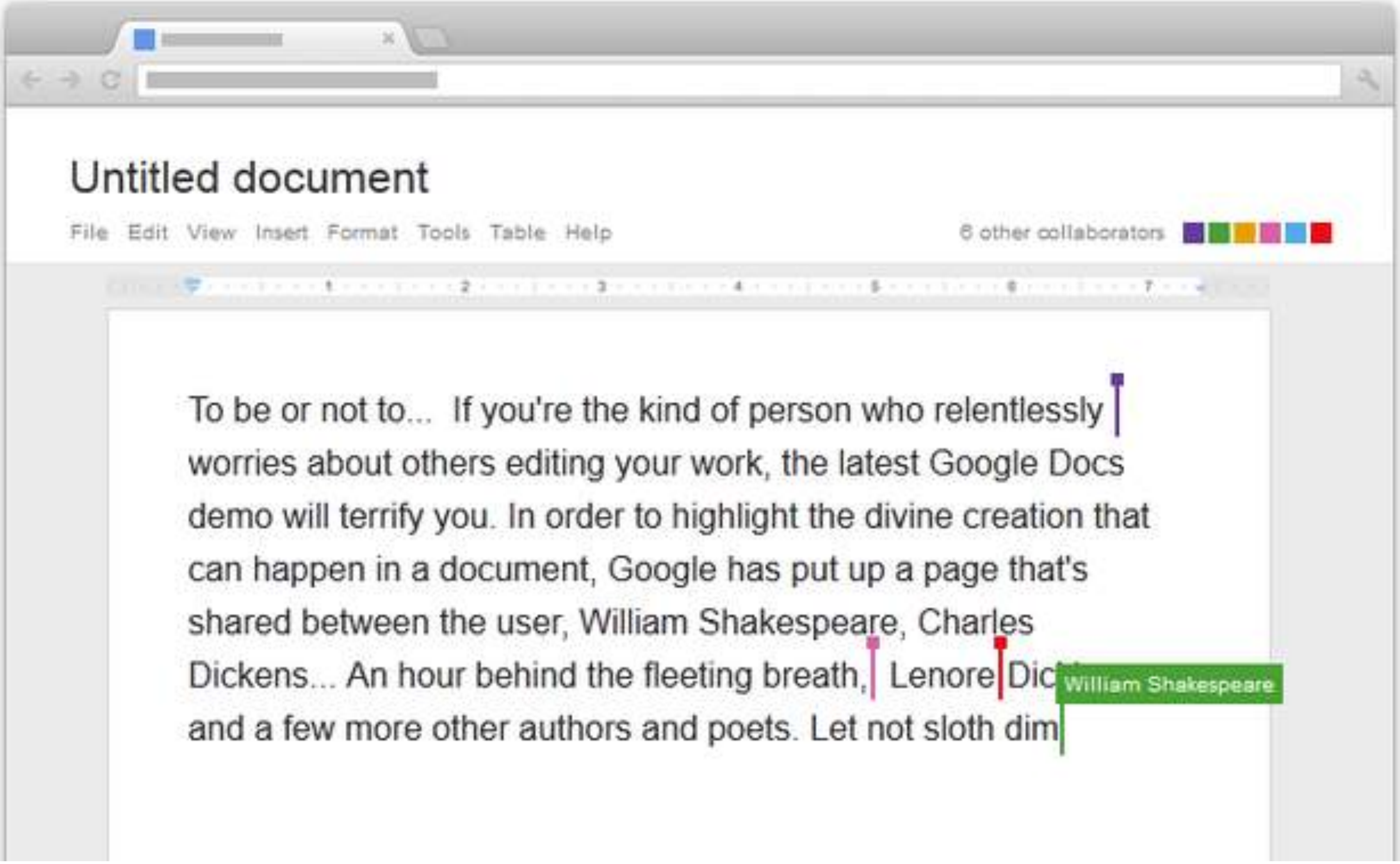
¿…?


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

¿Cómo hacer **convergencia** de estado?

¿Cómo hacer **convergencia** de estado si no podemos confiar en el **tiempo**?





Untitled document

To be or not to... If you're the kind of person who relentlessly worries about others editing your work, the latest Google Docs demo will terrify you. In order to highlight the divine creation that can happen in a document, Google has put up a page that's shared between the user, William Shakespeare, Charles Dickens... An hour behind the fleeting breath, Lenore Dic and a few more other authors and poets. Let not sloth dim

9° Foro de Cloud Computing:
Arquitectura Cloud

FOROS ISIS

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

CRDTs: Consistency without concurrency control

Mihai Leția — Nuno Preguiça — Marc Shapiro

9° Foro de Cloud Computing:
Arquitectura Cloud

FOROS ISIS

Conflict-free Replicated Data Types

Marc Shapiro, Nuno Preguiça, Carlos Baquero, Marek Zawirski

Universidad de los Andes
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas y Computación
AVANXO
amazon webservices
S4N
ORACLE
dotCloud beyond the cloud
intergrupo beyond...
grupodot
Microsoft

HAL
archives-ouvertes.fr

A comprehensive study of Convergent and Commutative Replicated Data Types

Marc Shapiro, Nuno Preguiça, Carlos Baquero, Marek Zawirski

Desde un punto de vista matemático, los CRDT no son más que

“**monotonic semilattice** data types”

Estructuras de datos

Asociativas

Conmutativas

Que conserva un “órden parcial” dado

Convergencia determinista.

🛒 🛒 🛒

🛒 |+| 🛒 |+| 🛒

🛒 |+| 🛒 |+| 🛒

Estado convergente **correcto**

Conclusiones.

Elasticidad por encima de escalabilidad.

La **elasticidad** en aplicaciones con estado no tiene **solución óptima**.

Hacer pruebas de este tipo de sistemas es **muy difícil**.

No le crean al que les diga que puede romper las reglas de la **física**.

Vamos a dejar nuestra implementación de **CRDT** como **OSS**.

- Weinstock, Charles., & Goodenough, John. (2006). ***On System Scalability*** (CMU/SEI-2006-TN-012). Retrieved June 16, 2015, from the Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=7887
- Frank McSherry and Michael Isard and Derek G. Murray (2015). ***Scalability! But at what COST?*** (15th Workshop on Hot Topics in Operating Systems (HotOS XV)). Retrieved May 2015, from USENIX Association, website: https://www.usenix.org/conference/hotos15/workshop-program/presentation/mcsherry
- Barbacci, Mario., Klein, Mark., Longstaff, Thomas., & Weinstock, Charles. (1995). ***Quality Attributes*** (CMU/SEI-95-TR-021). Retrieved June 16, 2015, from the Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12433
- Michael J. Fischer and Nancy A. Lynch and Michael S. Paterson (1985). ***Impossibility of Distributed Consensus with One Faulty Process***. Website: http://citeseerx.ist.psu.edu/viewdoc/summary;jsessionid=887BDE7DB319FED186AD8F8194653A8F?doi=10.1.1.13.6760
- ...

@ykiriki