Why trouble humans?
They do not care!

Know?
Understand?

Toni Cortes
Storage-system Research Group
Barcelona Supercomputing Center

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Storage System Research Group

An initial warning

- **I know "exascaling" is the hot topic today…**
  - Especially in a conference about parallelism

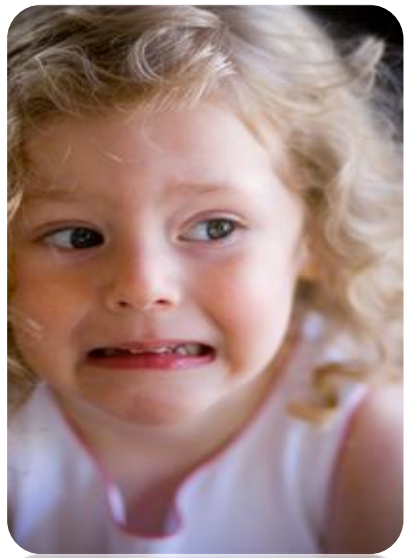
… but you have probably heard to much about it already!

- **I propose to talk about parallelism** (maybe "exascaling" too)

… but form a different point of view …

Does any of this seem familiar?

- **While using a "large" system**
  - How many times have you had to
    - Adapt your code or "behavior" to what the system expects?
      - Did you know why?
      - Did the motivation behind make any sense to you?
    - Tune parameters you do not understand?
  - *And … you are the experts!!!*



- **But, who is to blame?**
  - When solving a research challenge
    - How many *New* parameters have you added?
    - What is the increased complexity for users?

… But things may be changing!

First lesson to take home

- Parallel problems are hard …
  - … Exaflop systems are even harder …
    - … Should we put the pressure on users?
    - Happy users: best adopters of new technology
      - Programmers
      - System administrators
      - Scientists
      - Untrained users
      - …

No matter how hard your problem is, do not put the load on the user!

Storage System Research Group

But, can this be done?

- Did we ever think that a Cray2 (1st in Top 500 in 1985) …

… or any system in the Top500 till 1994 …

… could be as easy to use as an iPad in 2011?

- *Why not adding usability as a top requirement?*

Three examples for today

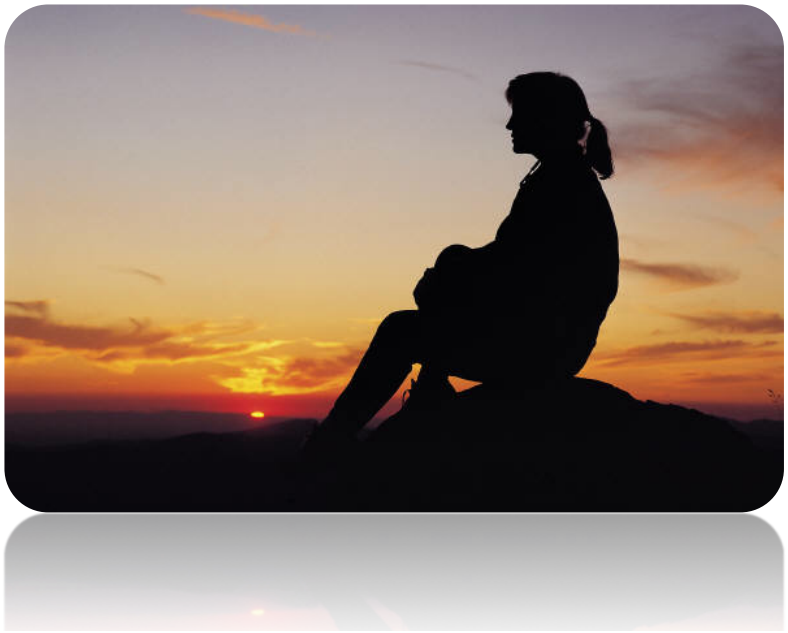
- **Do not put too many files in a single directory!**
  - But, I want to store a file per process!
    - And I have milions!
  - *Humans do not care or understand!*
- **The best disk scheduler depends on the load, device, …**

  - But, I need to configure one!
    - I always want the best I/O performance!
  - *Humans do not know!*
- **You should pick the fastest replica**
  - But how do I relate distance with latency?
    - And bandwidth?
  - *Humans do not know, understand, or care!*

Do not be fooled!

- **This is not a storage talk**
  - Though all examples are about storage
- **It is about designing usable parallel/distributed systems**
  - Do not forget: humans have to use them
- **The lessons to take home can be applied to any field**

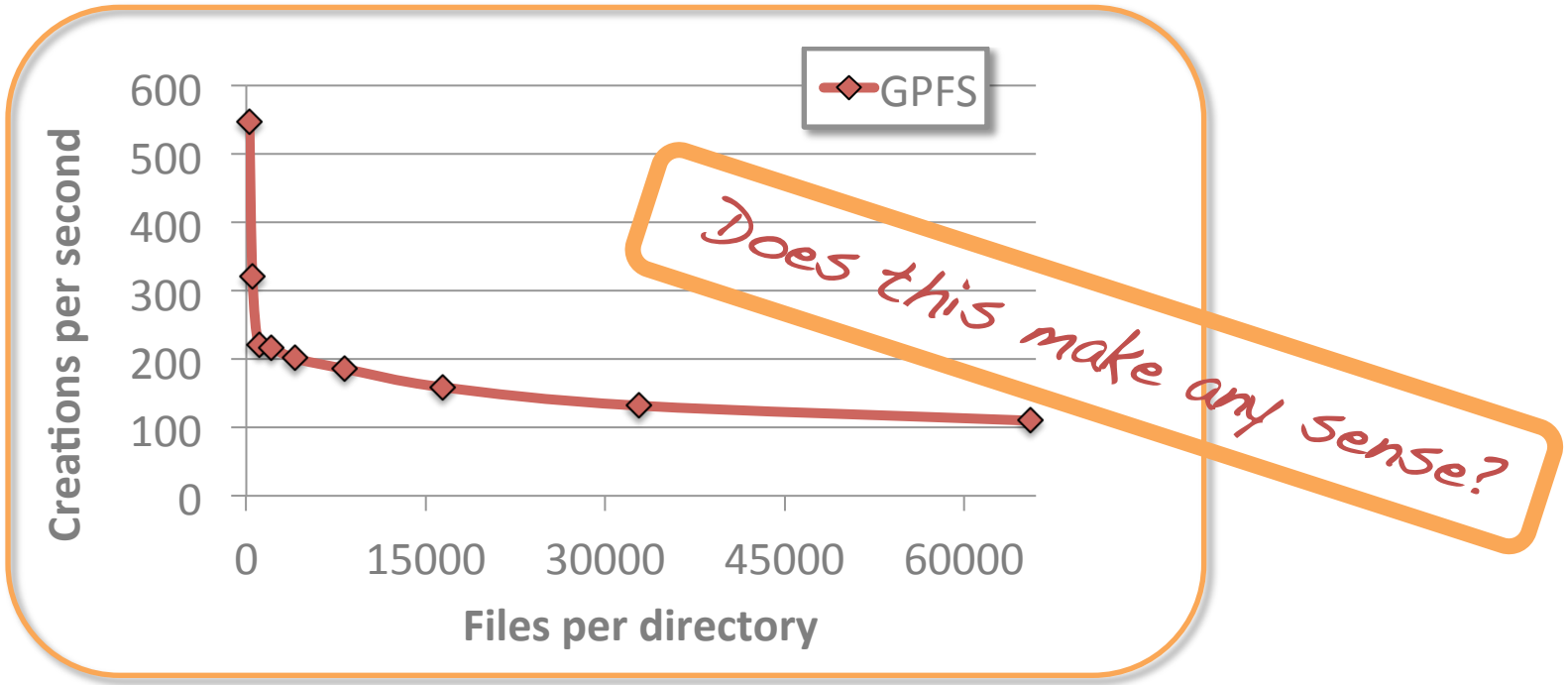
Should!



Example #1: Millions of files in a directory

The problem

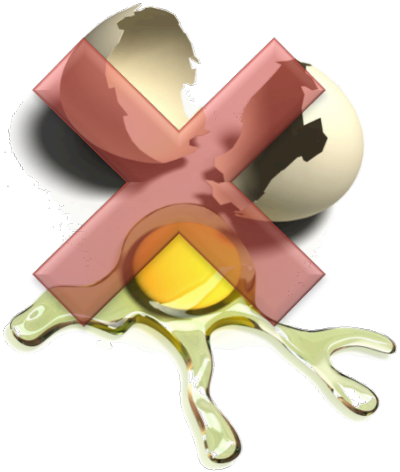
- If you create too many files in a directory …



- The solution system administrators give:
  - Distribute your files among several directories

Objective

- **Background**
  - Current system already do many things well
    - It makes no sense to build a new system (at least for us)
- **Objective:**
  - We want to solve the name space scalability
    - Directory locking (especially if many clients)
    - Internal structures
    - Basically ➔ directories with few files work!
  - Without breaking what is already working



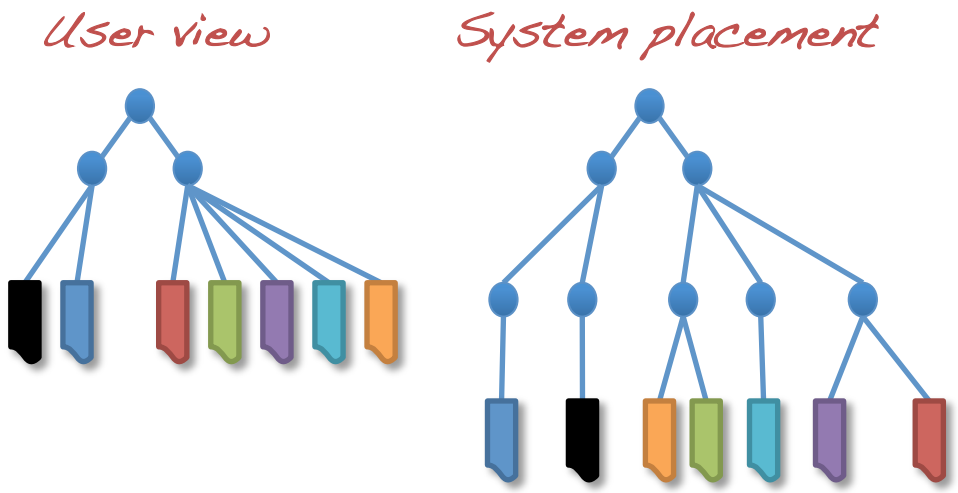
- **Idea**
  - Decouple the view users have from the system implementation
    - User view: many files in a single directory
    - System implementation: directories as the system likes them

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

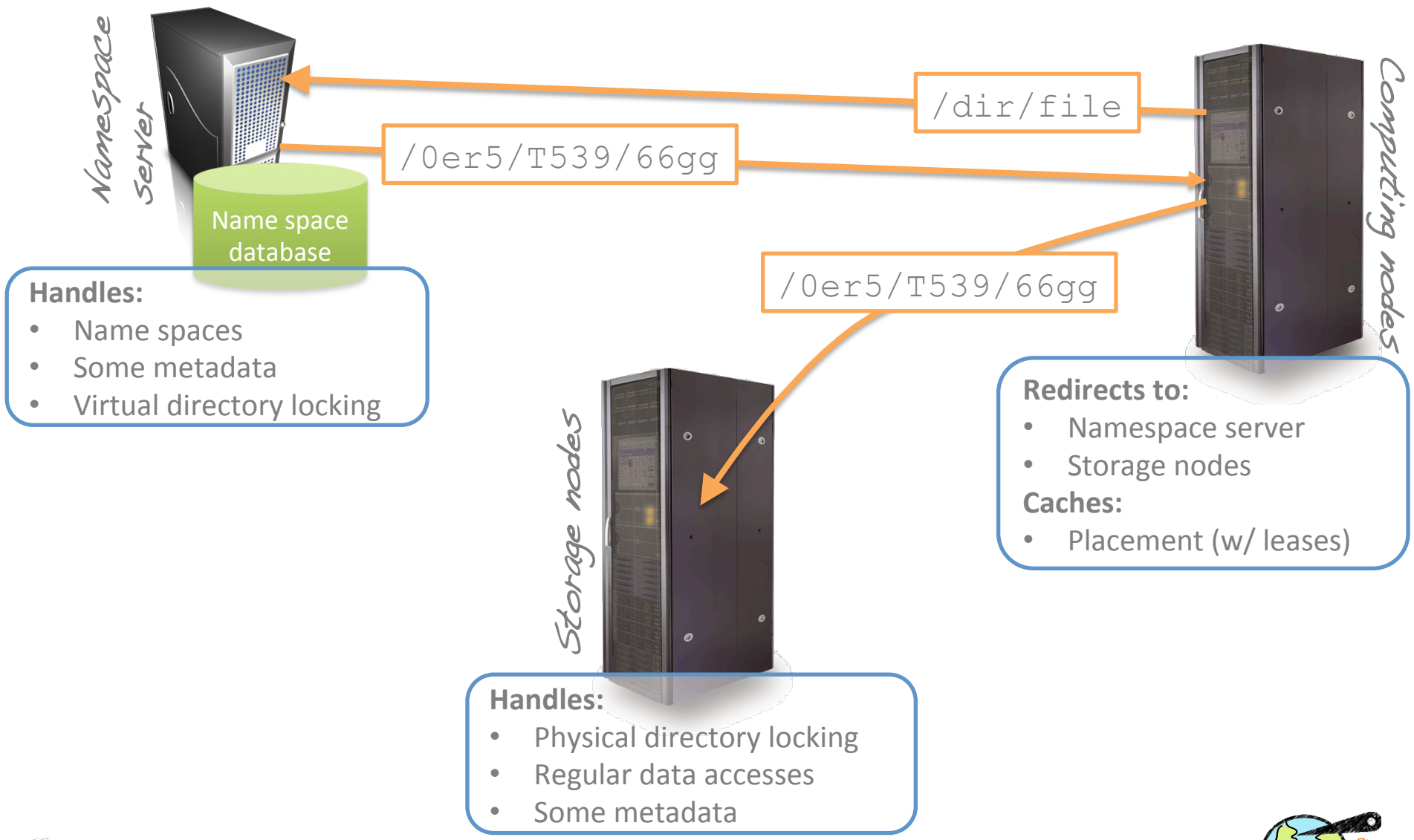
Storage System Research Group

User view vs. real placement

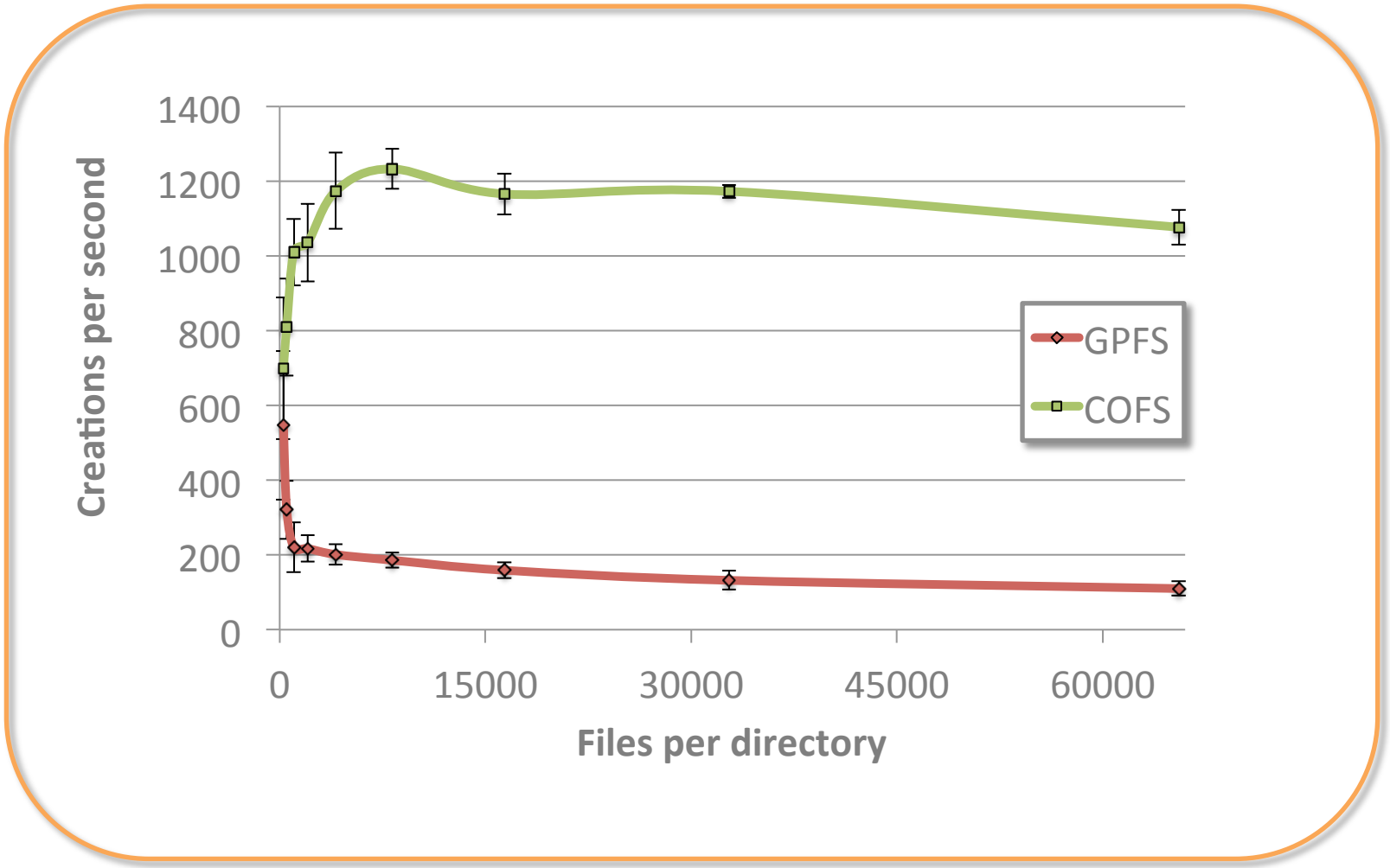
- **Proposed translation function**
  - Hash (client,
    Parent directory,
    Process,
    Random "bits")
    - Guarantees that a physical directory never has more than 512 files!
- **Example**
  - User file
    - /dir/fileA
  - Actual file
    - /0tr5/0j7a6/65fh



Architecture



Results


Creations per second
1400
1200
1000
800
600
400
200
0
0
15000
30000
45000
60000
Files per directory
GPFS
COFS

Lessons learnt

- Question assumptions regularly
  - *Who cared about metadata 20 years ago?*
- Decoupling user view from implementation is good
  - *Traditional mechanisms may not be the best*

- Future
  - Lessons learnt should be included in real systems

Example #2: Adaptable disk scheduler

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Storage System Research Group

The problem

- **Let's only compare CFQ and deadline**
  - Linux kernel read 32 processes works 25% better with **deadline**
  - … but IO-R 32 processes works 60% better with **CFQ**

Depends on the application!

  - Linux kernel read
    - When running with 8 process, **CFQ** is 25% better
    - When running with 32 processes, **deadline** is 15% better

Depends on the number of processes!

  - TAC benchmark with 4 processes
    - Using a Seagate ST3250310NS, **CFQ** is 20% better
    - Using a Seagate ST3500630AS, **Deadline** is 20% better

Depends on the disk used!

How can I know?

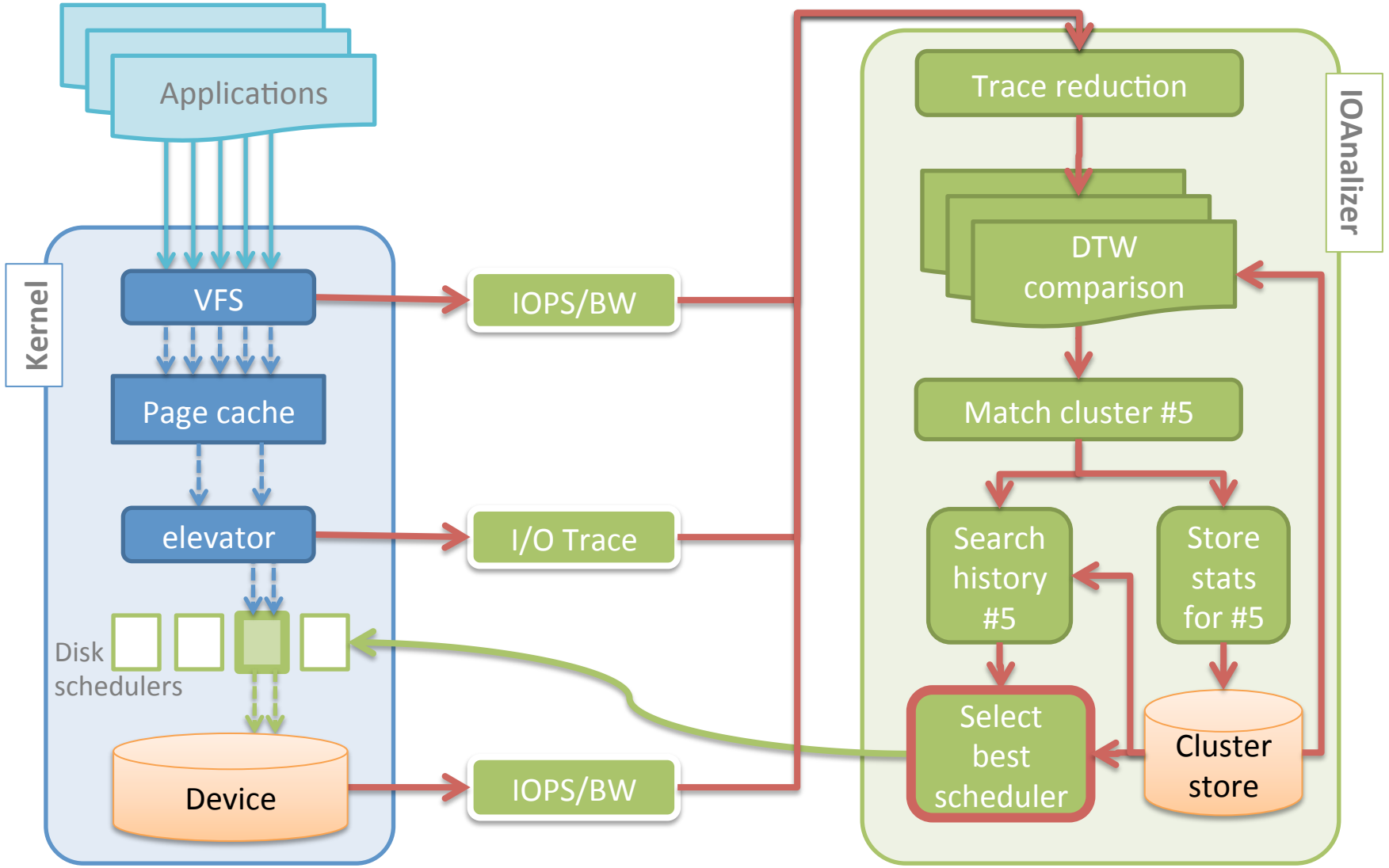
Objective

- **Background**
  - Current systems have several disk schedulers
    - Their performance depends on the load, devices, …
    - Very difficult to chose (and to tune their parameters)
  - New architectures have many cores
    - They will not be always used
    - Cores will be like memory
      - Who uses all memory?
- **Objective:**
  - A mechanism that, on-line, selects the best scheduler
    - … and its parameters
- **Idea**
  - Use available cores to perform time-consuming analysis

Cluster of requests

- **Assumption: similar patterns need same disk scheduler**
- **From access patterns to clusters**
  - Capture the trace of I/O requests
  - Reduce the trace (we want to support 1Milion ops per second)
    - Divide the trace in 1000 buckets
      - Select the max or min sector jump randomly
    - Include IOPs
  - Cluster reduced patterns using FastDTW
- **Using clusters**
  - Learn the best scheduler per cluster
    - Current version: by trial and error
  - Learn sequences of clusters
  - When a cluster is detected, change to best scheduler for next one

Architecture

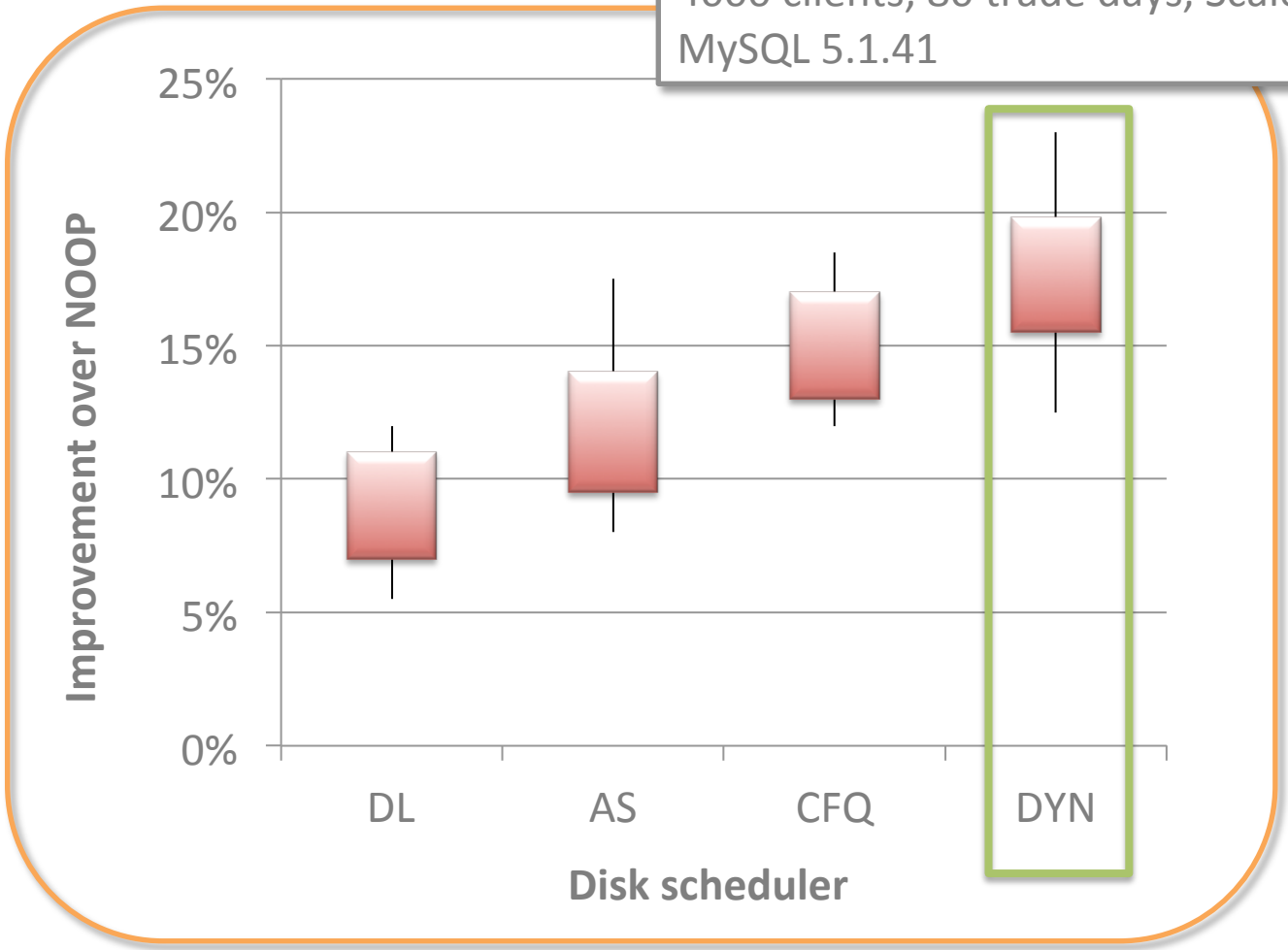


Why multi cores?

- **These operations *are* quite time consuming**
  - Reducing traces
    - Needs to be done fast
  - FastDWT
    - Quite time consuming, but not in critical path
- **These operation *will be* quite time consuming**
  - Use simulations to predict the behavior of an scheduler
    - We could try many more parameters
    - In our roadmap for the near future

Results using TCP-E

Amazon/ebay load
4000 clients, 80 trade days, Scale factor 500
MySQL 5.1.41



Lessons learnt

- We will have plenty of cores in future systems
  - *Why not use them to improve system software?*
    - No all of them will used anyway!
- Many things were not possible in the past
  - *Can we revisit them?*

Example #3: Automatic location of replicas

The problem

How can a user guess?

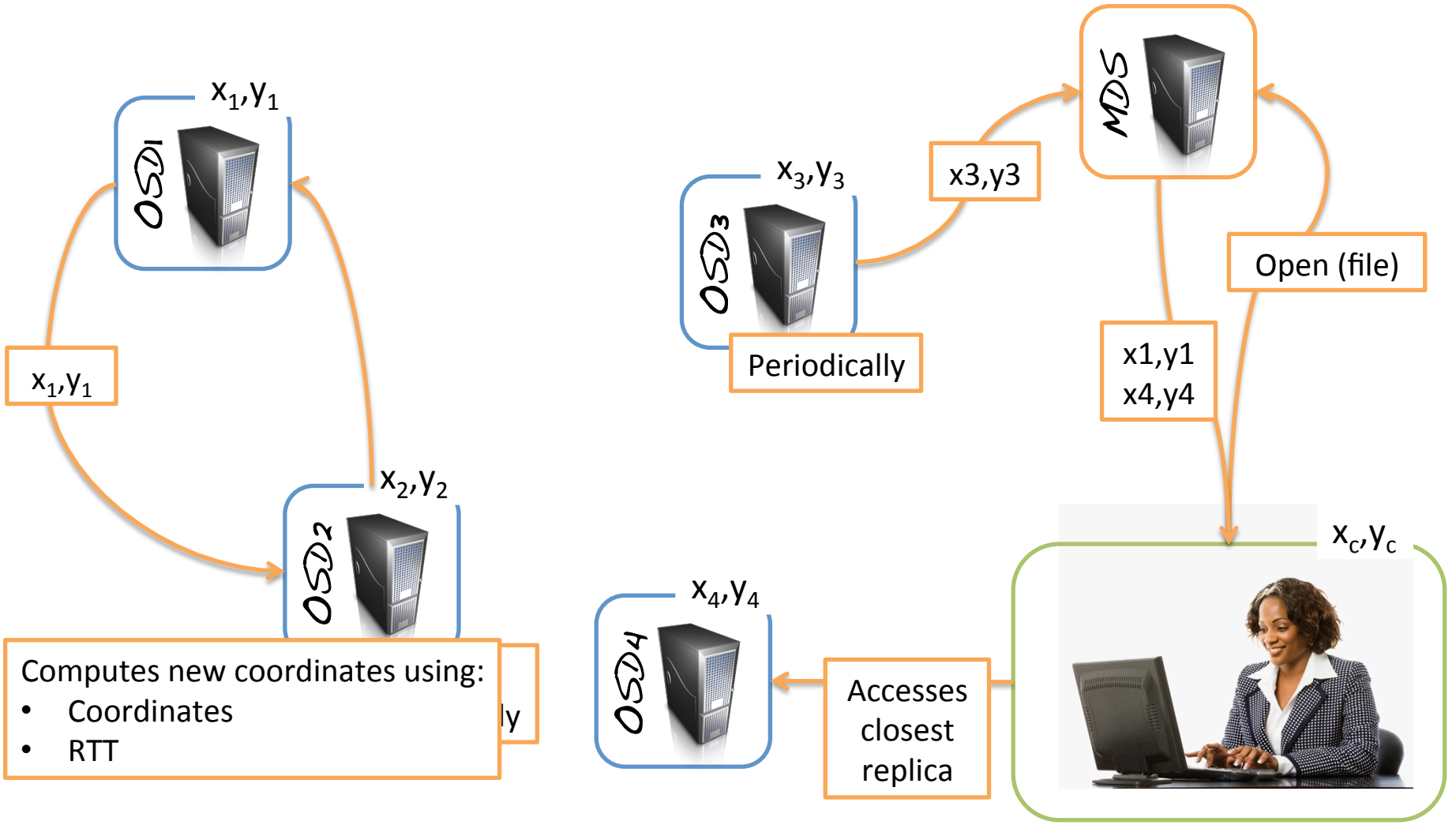
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Storage System Research Group

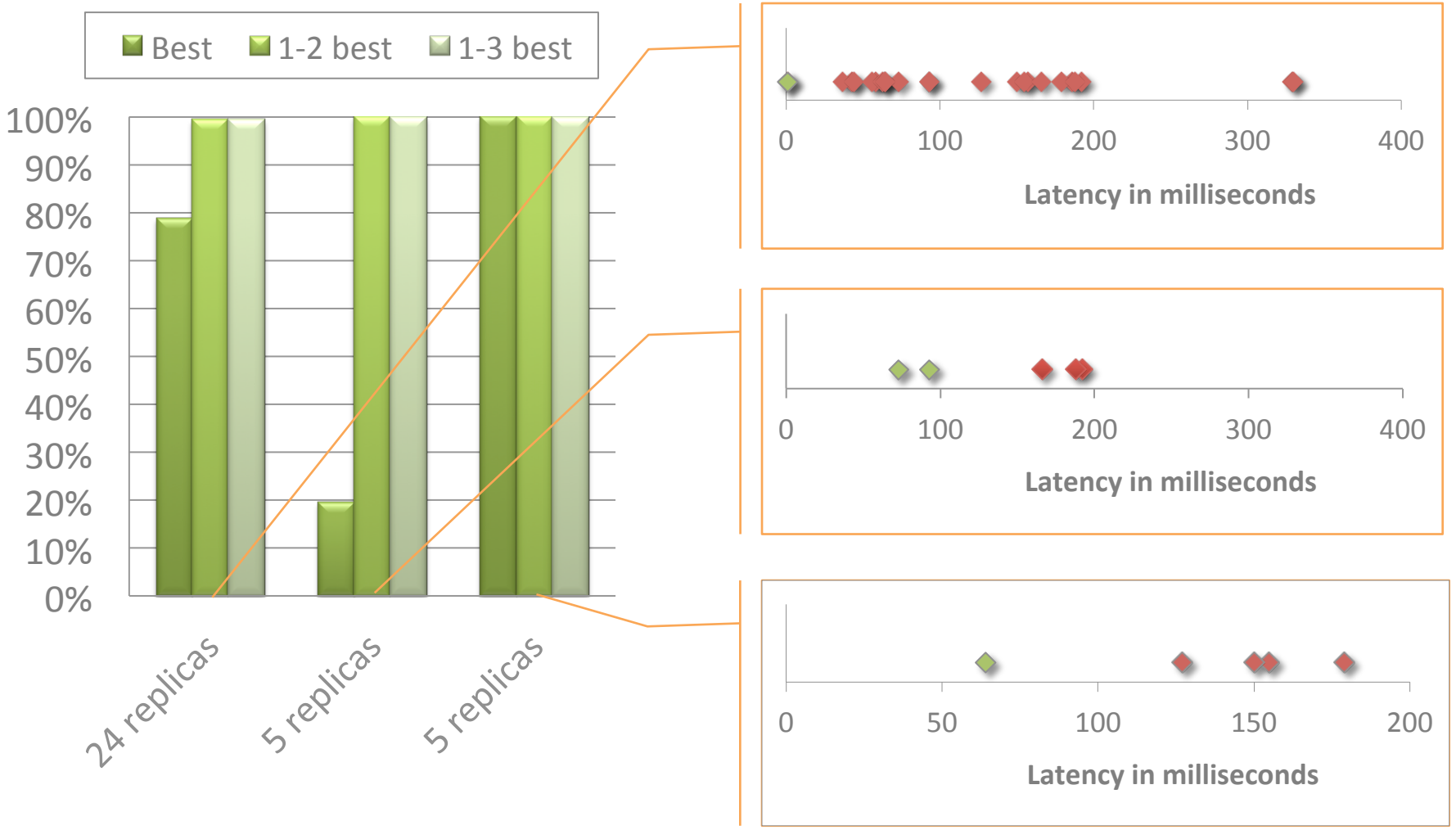
Objective

- **Background**
  - Vivaldi
    - Mechanisms to map latency distances in an Euclidean space
    - Dynamically adapts to changes in latency
  - Not the only mechanism
- **Objective**
  - Apply this idea to XtreemFS/XtreemOS
    - When using a file
      - Select the closest replica to a client (latency wise)
    - When launching a job
      - Select resources close to replicas (latency wise)
- **Idea**
  - Map clients and OSDs in a 2D space
    - Use Euclidean distance

Architecture



Results


Best
1-2 best
1-3 best
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
24 replicas
5 replicas
5 replicas
0
100
200
300
400
Latency in milliseconds
0
100
200
300
400
Latency in milliseconds
0
50
100
150
200
Latency in milliseconds

Lessons learnt

- Very simple ideas can work fairly well
  - How much accuracy can you lose to make it …
    - Usable?
    - Scalable?

General conclusions

Wish #1

- **Conclusion #1**
  - I hope I cannot reuse these slides in 10 years
- **Other conclusions**
  - Question assumptions regularly
  - Traditional mechanisms may not be the best
  - Use available computing power to revisit old ideas
  - Approximate solutions are good in many cases
    - And they are usable!
    - And they scale!

Thanks to …

- Ernest Artiaga (BSC)
- Jacobo Giralt (BSC)
- Juan González (BSC)
- Pilar González (U. Murcia)
- Ramón Nou (BSC)
- Jesús Malo (now at Quiterian)
- Joanthan Martí (BSC)
- Juan Piernas (U. Murcia)

- And the **storage-system research group at BSC** in general

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Storage System Research Group

More information

Storage system research group
Barcelona Supercomputing Center

www.bsc.es/StorageSystems

IOlanes
European funded project

www.iolanes.eu

XtreemOS
European funded project

www.xtreemos.eu