KERLABS

Kerrighed 2.4

Renaud Lottiaux
Kerlabs

Renaud.Lottiaux@kerlabs.com

Outline

The Kerrighed OS

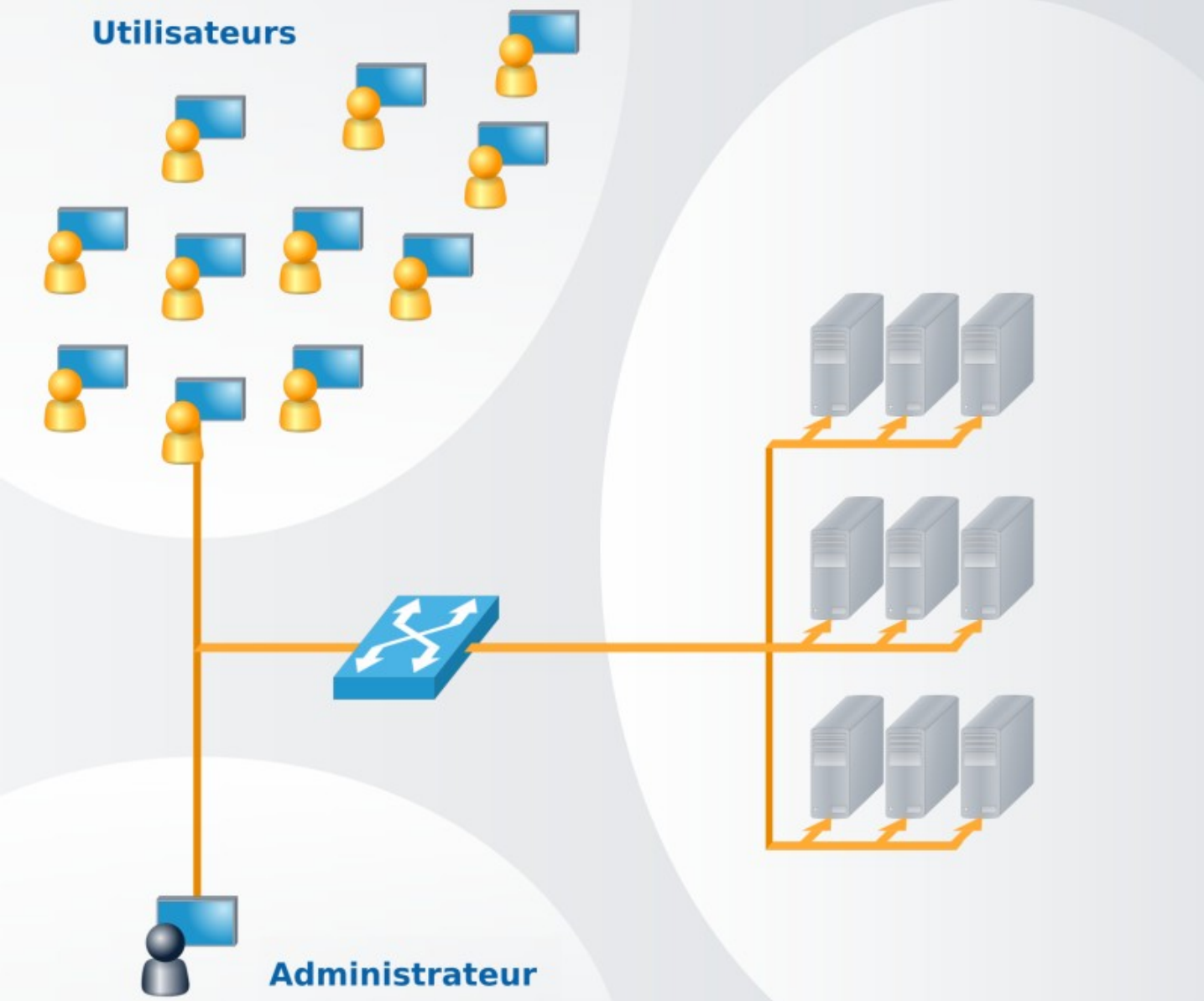
Memory injection

Configurable scheduler

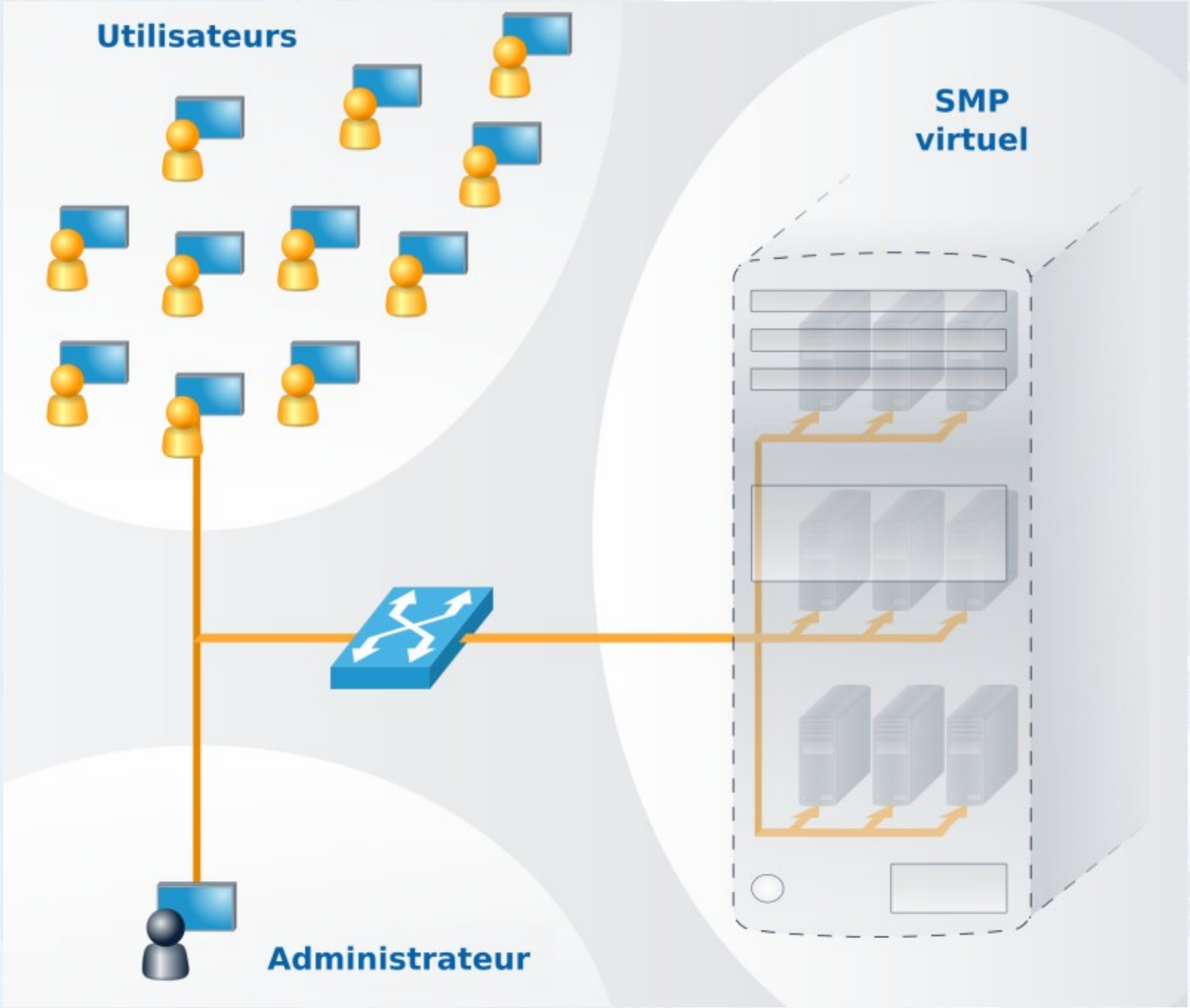
Application check-pointing

Power saving in Kerrighed clusters

Kerrighed : A Single System Image OS



Kerrighed : A Single System Image OS



Kerrighed

Project started at INRIA in 1999

Collaboration with University of Rennes 1 and EDF

3 Ph.Ds

Many engineers and interns

30 man.year of R&D

2005 : Creation of Kerlabs

INRIA Spin-off

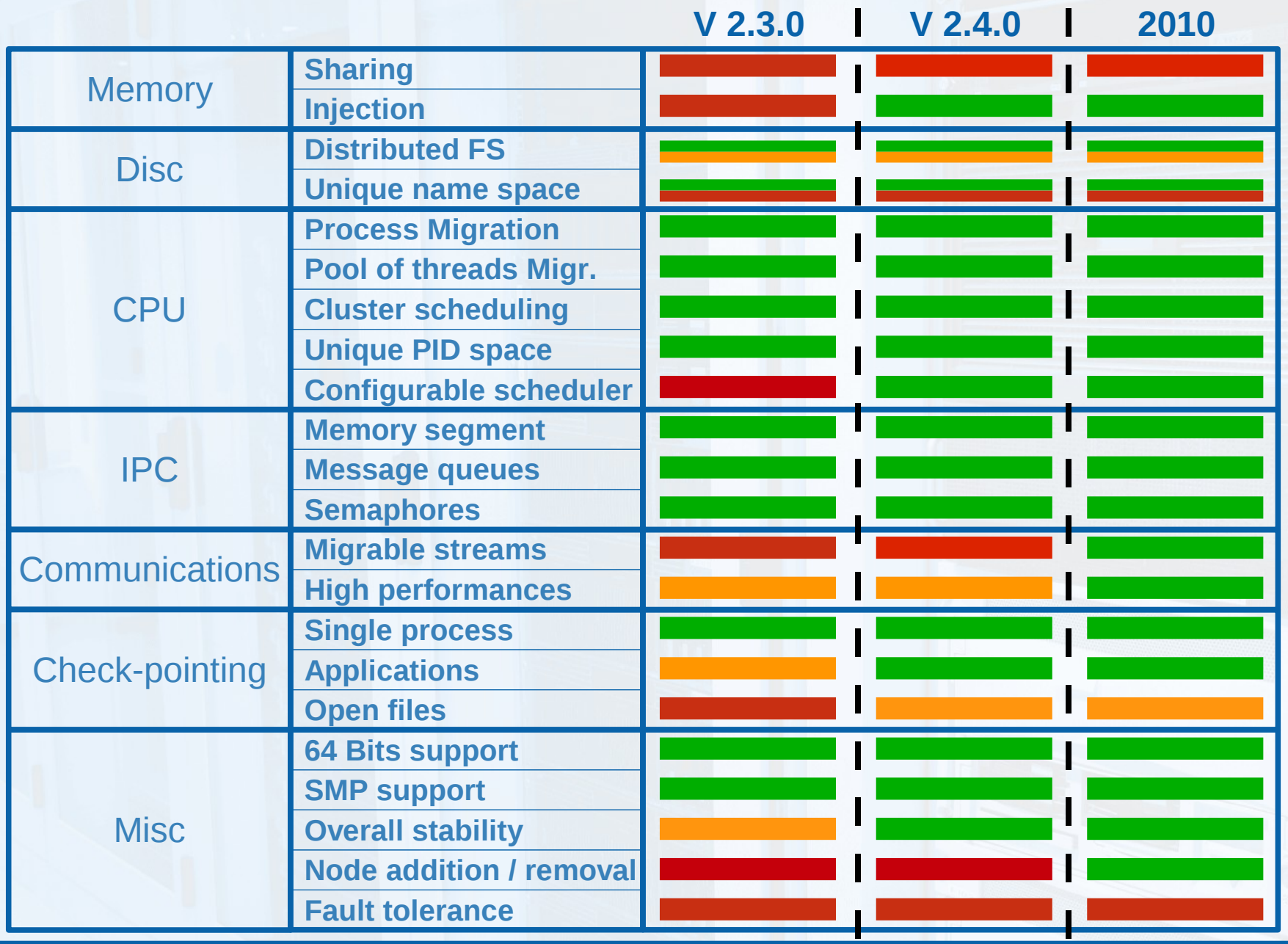
May 2005 – September 2006 : gestation of the company

October 2006 : actual creation of the company

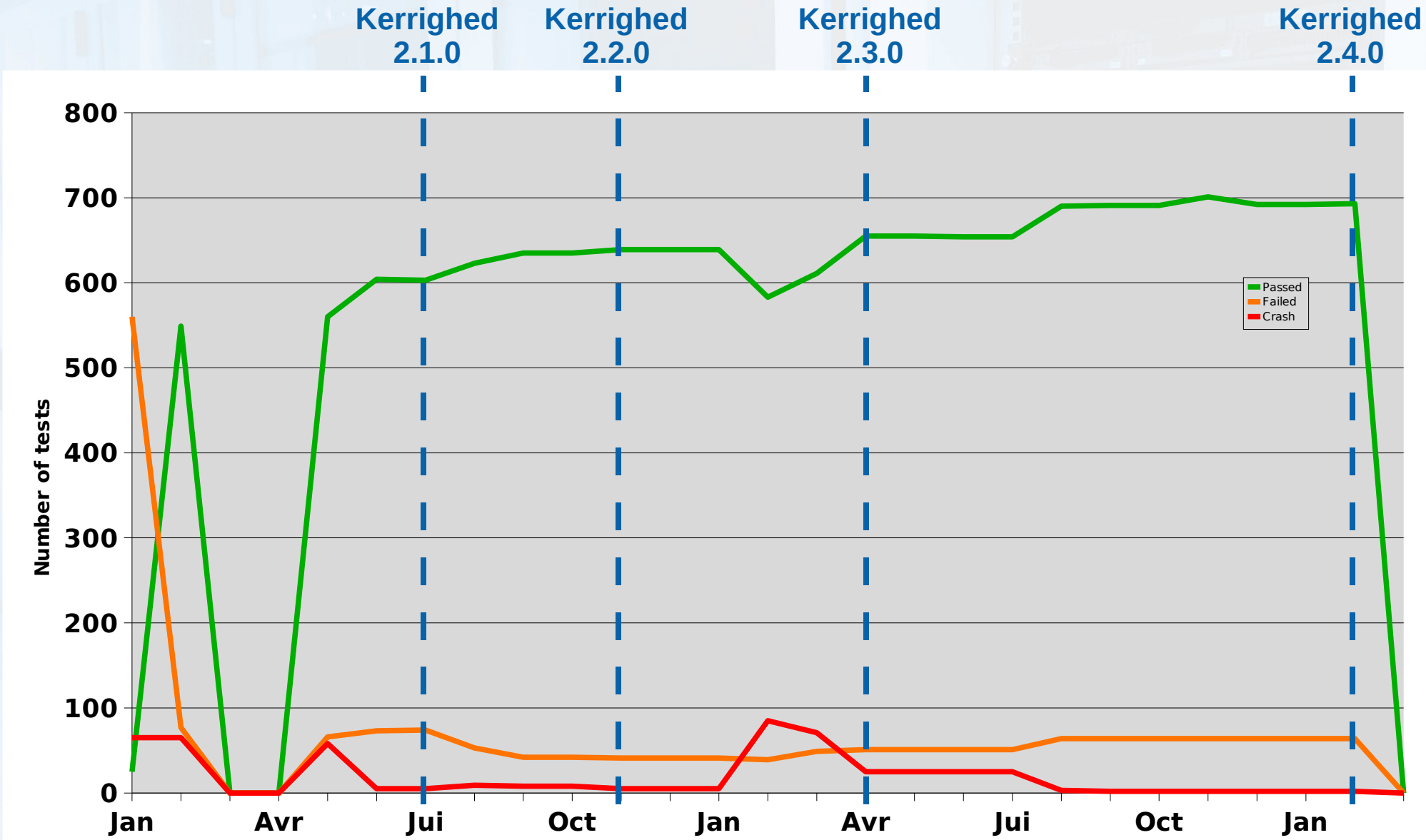
Goal : industrialize Kerrighed

Since 2005, Kerrighed is mainly developed by Kerlabs

New features in version 2.4.0



Evolution of LTP tests



Outline

The Kerrighed OS

Memory injection

Configuration scheduler

Application check-pointing

Power saving in Kerrighed clusters

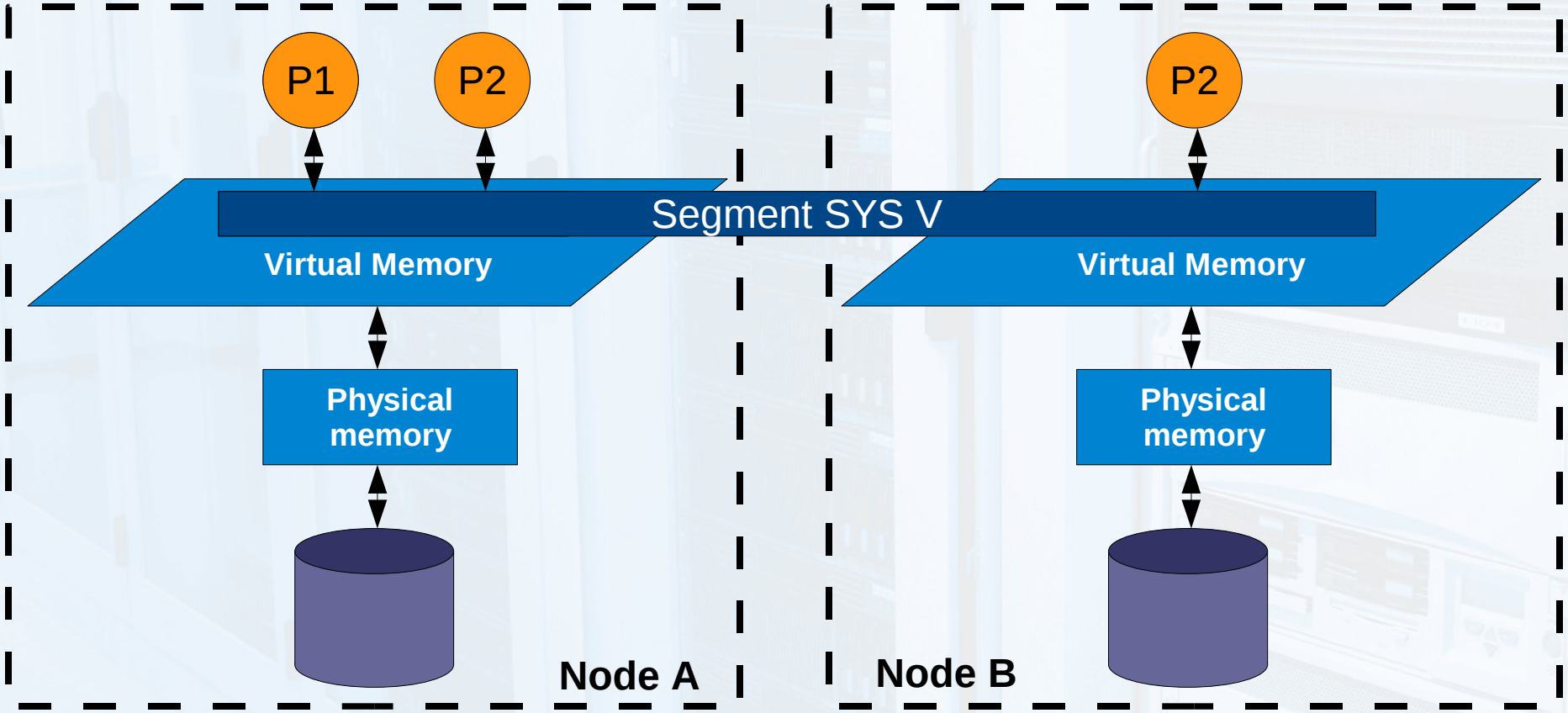
Memory sharing in Kerrighed (1)

Shared virtual memory

N processes share the same virtual memory area

System V segments

Thread (currently disabled)



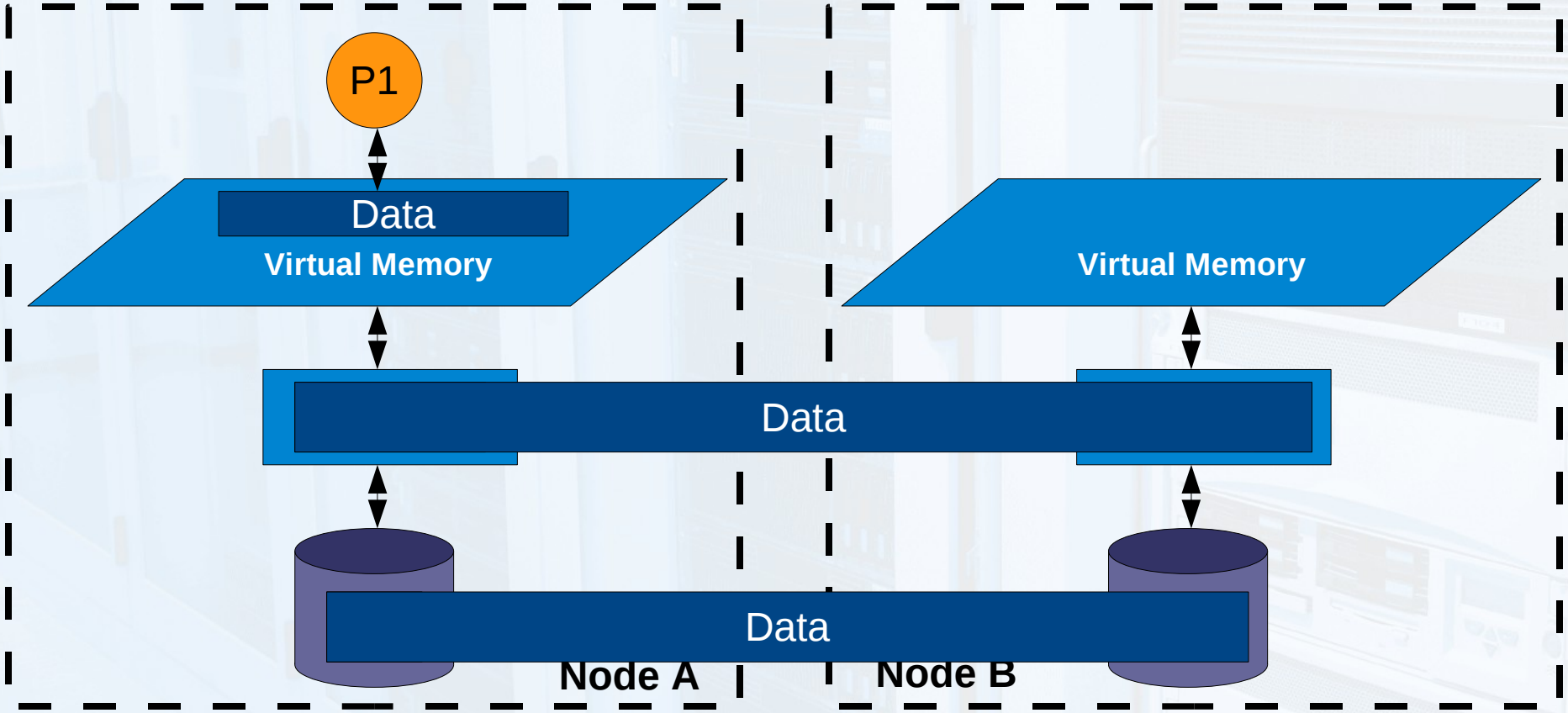
Memory sharing in Kerrighed (2)

Memory injection (new in 2.4)

1 process uses the memory of N nodes

Add a new memory layer

Local Memory -> Remote Memory -> Disc



Understanding the memory injection

Regular system : a 2 level hierarchy

Physical memory

Capacity : 2 GB

Throughput : ~5 GB / s

Latency : ~60 ns

Hard drive

Capacity : 80 GB

Throughput : ~5 à 50 MB / s (100 to 1 000 times < memory)

Latency : ~5 ms (100 000 times > memory)

Easy to understand how the swap kill performances !

Understanding the memory injection

Injection add a new level to the memory hierarchy

Remote memory through Gb Ethernet network

Throughput : ~120 MB / s

40 times < memory

2,5 to 20 times > disc

Latency : ~30 us

500 times > memory

200 times < disc

Remote memory through Infiniband network

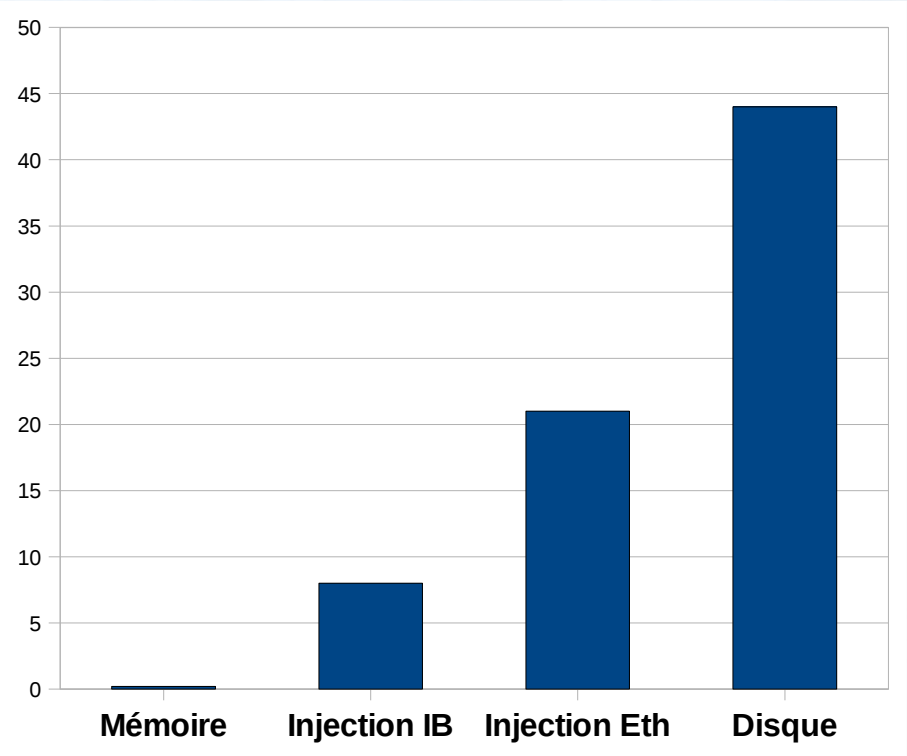
Throughput : ~2,5 GB / s

2 times < memory

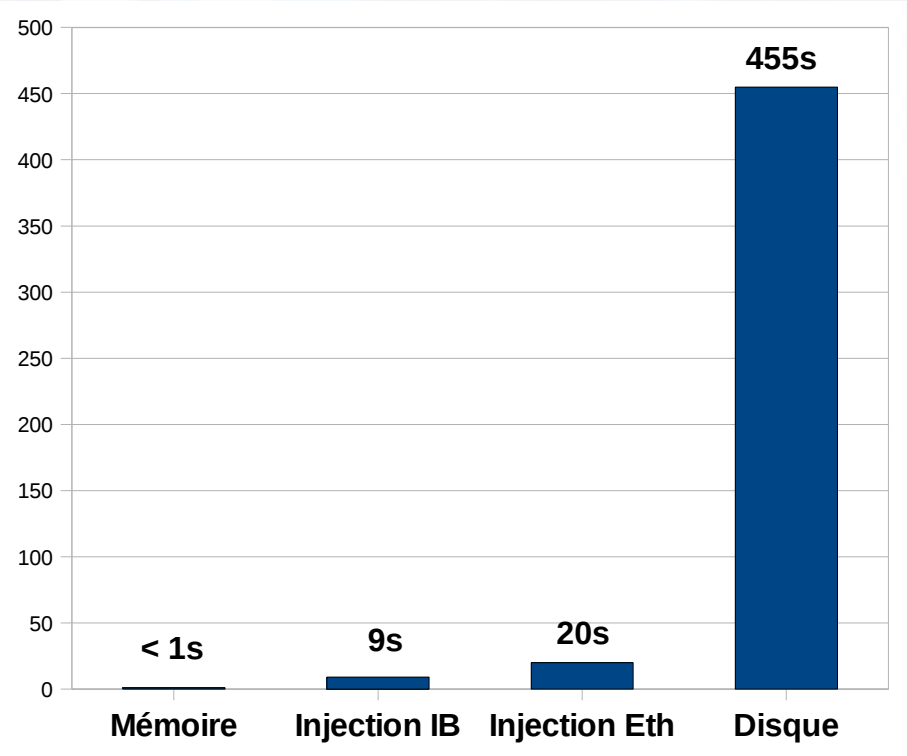
Latency : ~2 us

33 times > memory

Performance (1)

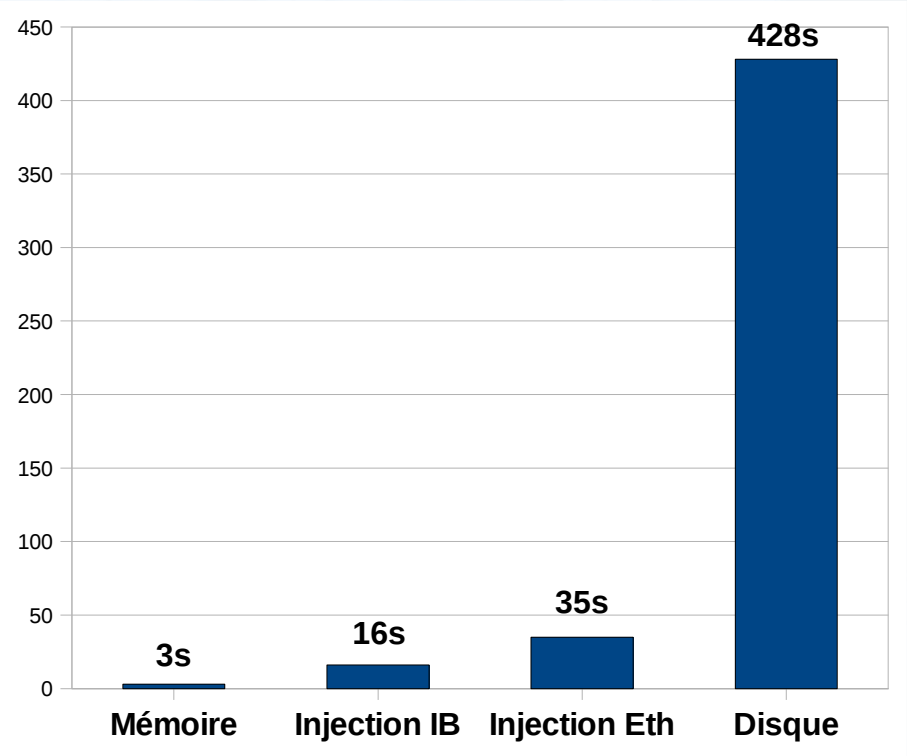


Sequential memory access

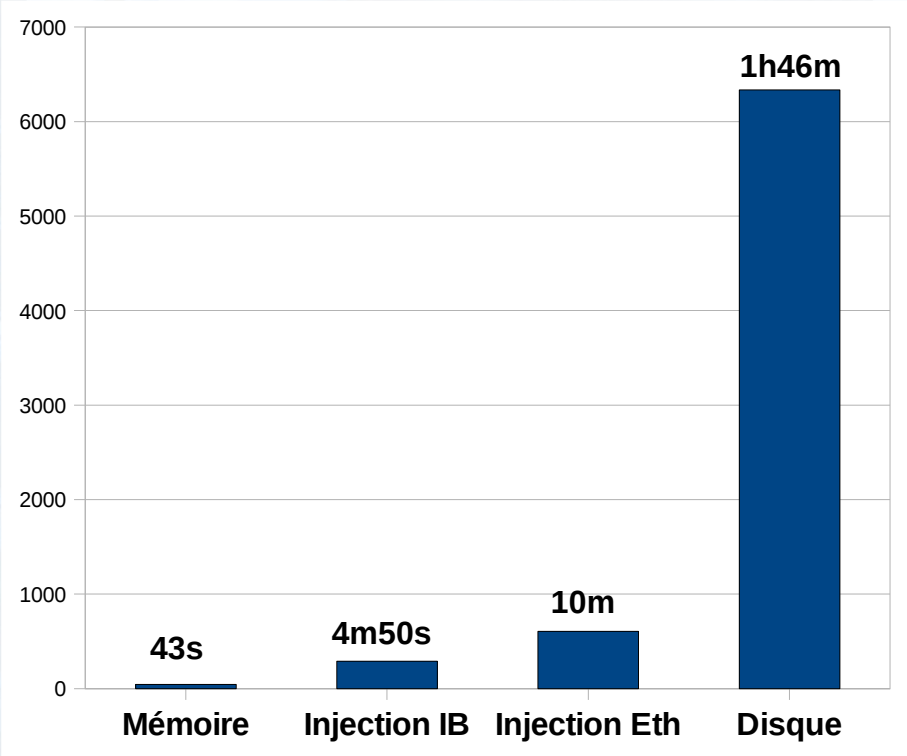


Random memory access

Performance (2)



Matrix transpose



Mechanical code

Optimization

Many optimization are in development

Prefetching

Page aggregation

Access pattern detection

0 copy communication

Use of RDMA

Expected results

Gb Ethernet : speed-up of 2

Infiniband : speed-up of 5

Outline

The Kerrighed OS

Memory Injection

Configurable scheduler

Application Check-pointing

Power saving in Kerrighed clusters

Global process scheduler

Goam : balance the CPU load on a cluster

Move processes from one node to another

Automatic migration

Several possible policies

Task injection

Task stealing

...

Several criterion

CPU load

Memory load

CPU temperature

...

Issues

Scheduler hard-coded in the system

One “generic” algorithm

One scheduling criterion

Most of the time: balance CPU load

Many limitations

Scheduler can make scheduling mistakes

The user may wish to control process placement

New scheduling criterion can appear

To change the policy

In the best case: change the code in the kernel

In the worst case: impossible !

Kerrighed Capabilities

A first level of control in Kerrighed

Capabilities

Control the behavior of the scheduler

Enable / disable scheduler features

Granularity : process

Capabilities are inherited between processes

2 capabilities

DISTANT_FORK

Enable / disable task placement at start-up

Useful for short-running tasks

CAN_MIGRATE

Enable / disable task migration

Useful for long-running tasks

New feature : configurable scheduler

Goal: modify the process scheduler on a running system

Adapt the policy to application needs

Create new polocies

Disc affinity

Network afinity (MPI tasks for instance)

Disc load balancing

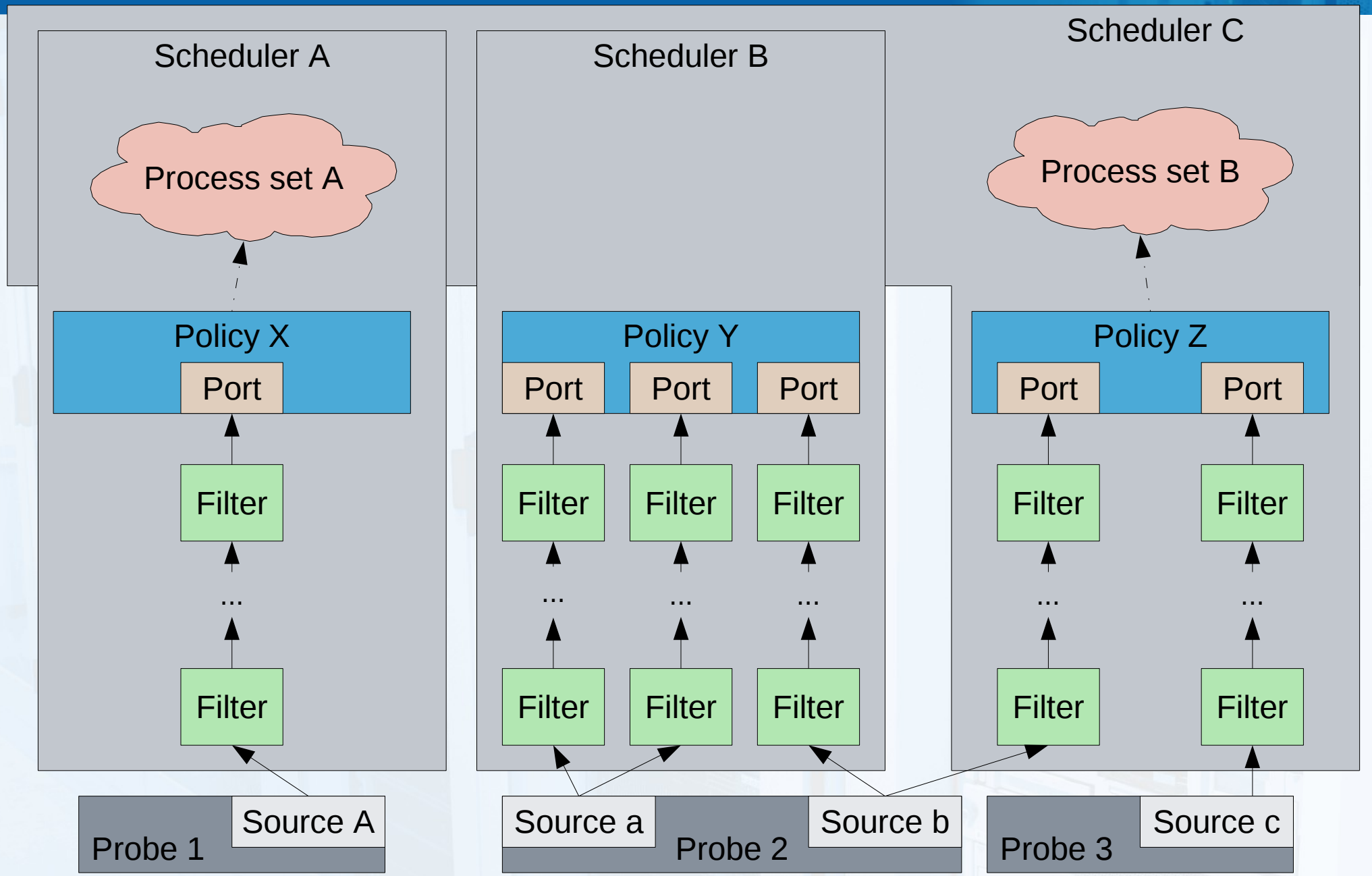
Avoid creation of “hot spot”

Etc...

Control tasks managed by schedulers

Create a scheduler hierarchy

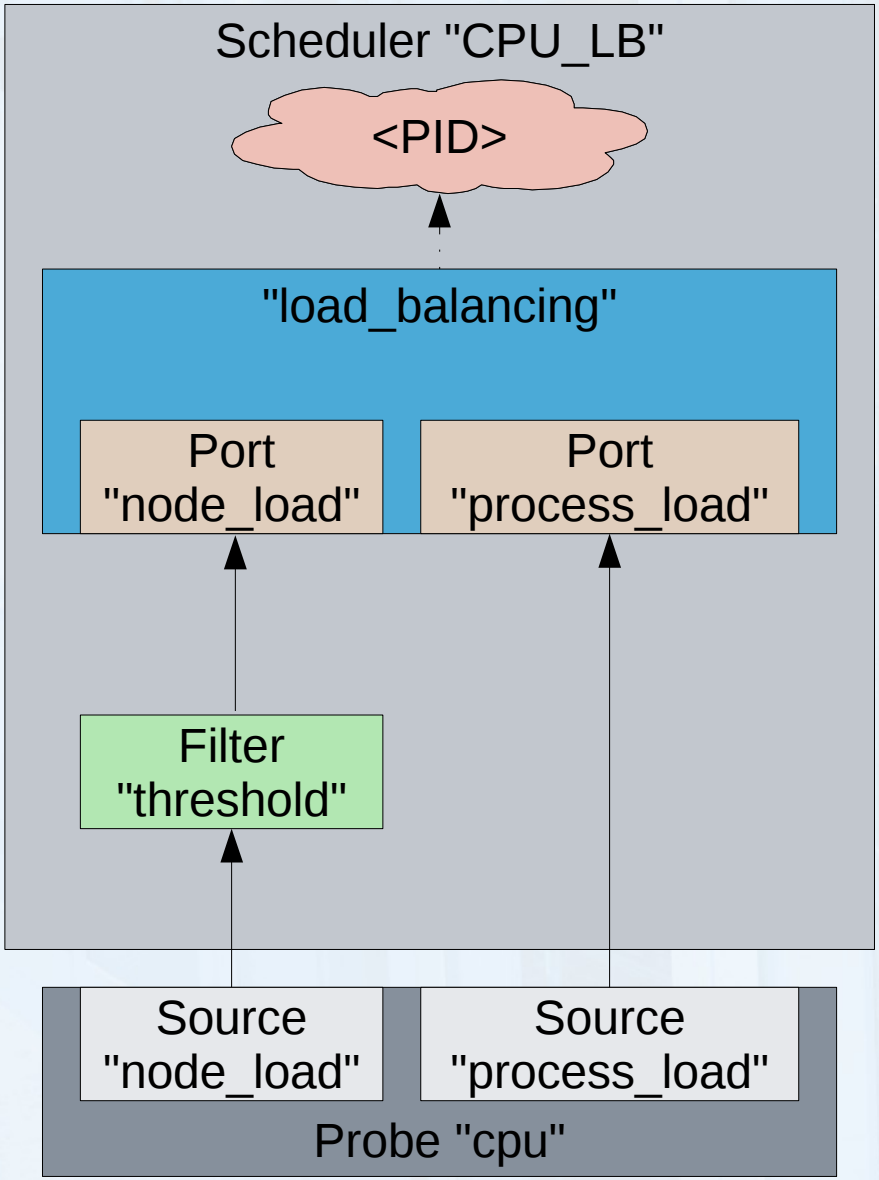
Configurable scheduler architecture

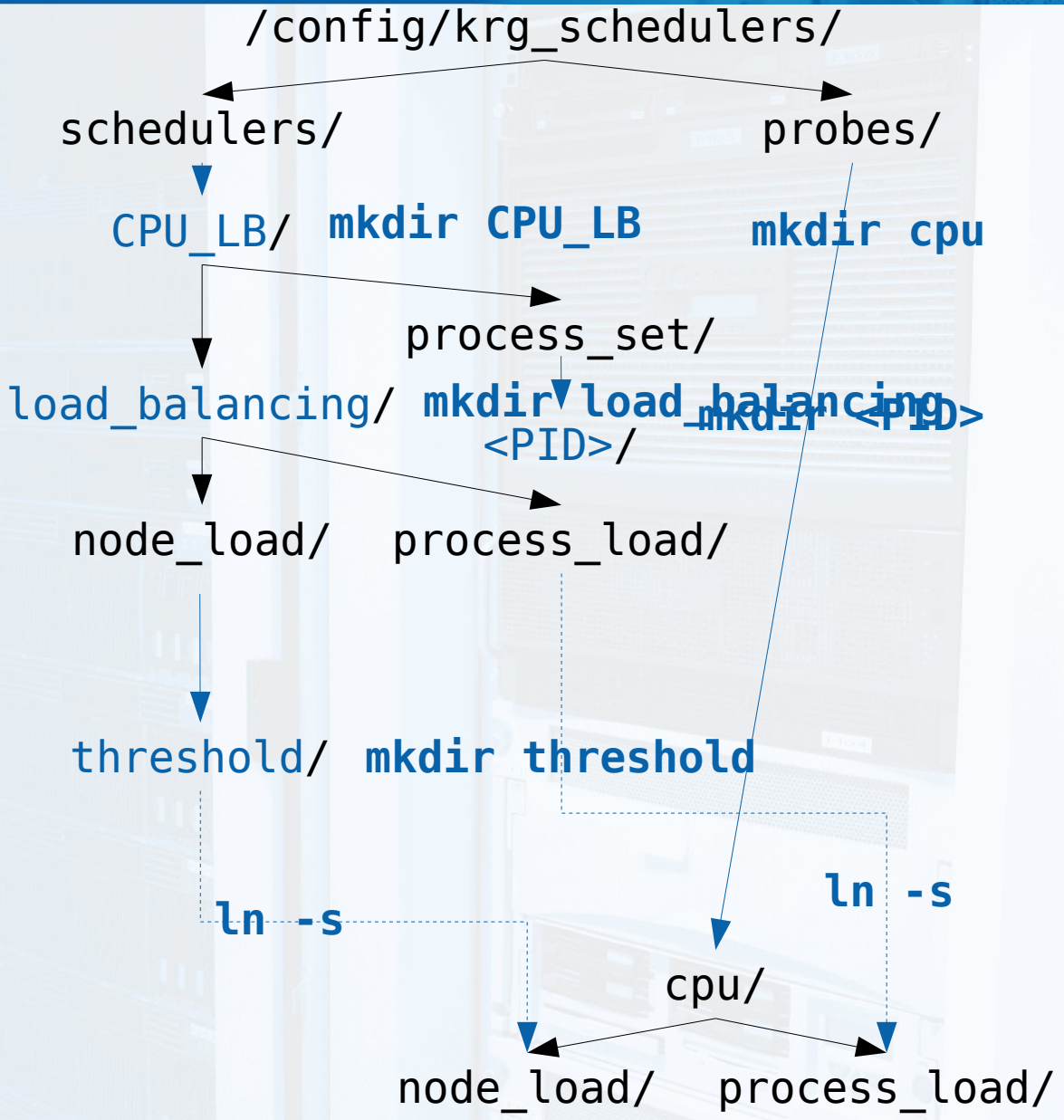


Features

- User interface
  - Configuration using Config-FS
  - Cluster wide configuration
- Kernel level
  - Module
    - Probes, filters, policies
  - Dedicated API to plug scheduler components

Configuration using Config-FS





Some “real-life” examples

Token scheduler

A USB key holds token on each cluster node

Non uniform token consumption

Goal

Balance token consumption

Inform cluster admin when a key passes below a token threshold

Priorities in a mailing campaign

Several campaigns to send simultaneously

Each mailing campaign has a given priority

Goal

Send as fast as possible (using more resources) campaigns with highest priority

Outline

The Kerrighed OS

Memory injection

Configurable scheduler

Application check-pointing

Power saving in Kerrighed clusters

Process check-pointing

Save a snapshot of a running process

Enable to restart a process from the last snapshot

User interface: 2 shell commands

Checkpoint <pid>

```
$ checkpoint 4195676
Identifier: 4195675
Version: 1
Description: No description
Date: Tue Mar 31 14:34:00 2009
```

Restart <application identifier> <version number>

```
$ restart 4195675 1
Restarting application 4195675 (v1) ...
Done
```

Application check-pointing

An application =

A set of processes

A set of « links » between processes

Father / sons filiation

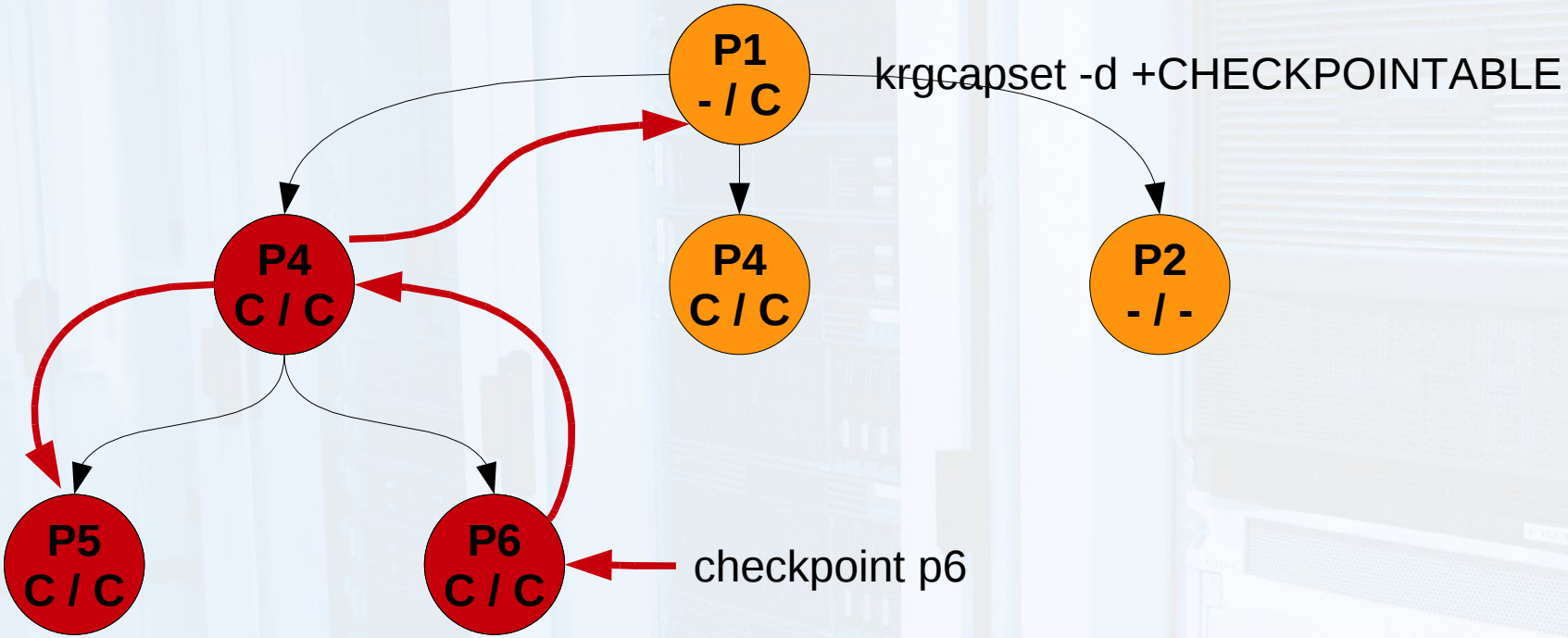
Checkpoint frontier

By default, a process cannot be check-pointed

Capability « CHECKPOINTABLE » must be set to the process

Make it possible to control the check-point frontier

Check-point frontier



Process check-point

```
$ checkpoint 4195676
Freezing application in which process 4195676 is involved...
Checkpointing application in which process 4195676 is involved...
Identifier: 4195675
Version: 1
Description: No description
Date: Tue Mar 31 14:34:00 2009
Unfreezing application in which process 4195676 is involved...
```

```
$ restart 4195676 1
Restarting application 4195675 (v1) ...
Done
```

Corner cases

Case of open files

Open read-only: nothing to do

Open read-write: external copy

```
$ checkpoint -f 4195676
Freezing application in which process 4195676 is involed...
$ cp ~/data.file /backup
$ checkpoint -c 4195676
Checkpointing application in which process 4195676 is involved...
Identifier: 4195675
Version: 1
Description: No description
Date: Tue Mar 31 14:34:00 2009
$ checkpoint -u 4195676
Unfreezing application in which process 4195676 is involed...
```

```
$ cp /backup/data.file ~/
$ restart 4195676 1
Restarting application 4195675 (v1) ...
Done
```

Future developments

Checkpoint of IPC system V objects

Memory segments

Semaphores

Message queues

Incremental checkpointing

Only save data modified between 2 check-points

Call-back

Inform the application about check-point events

Outline

The Kerrighed OS

Memory injection

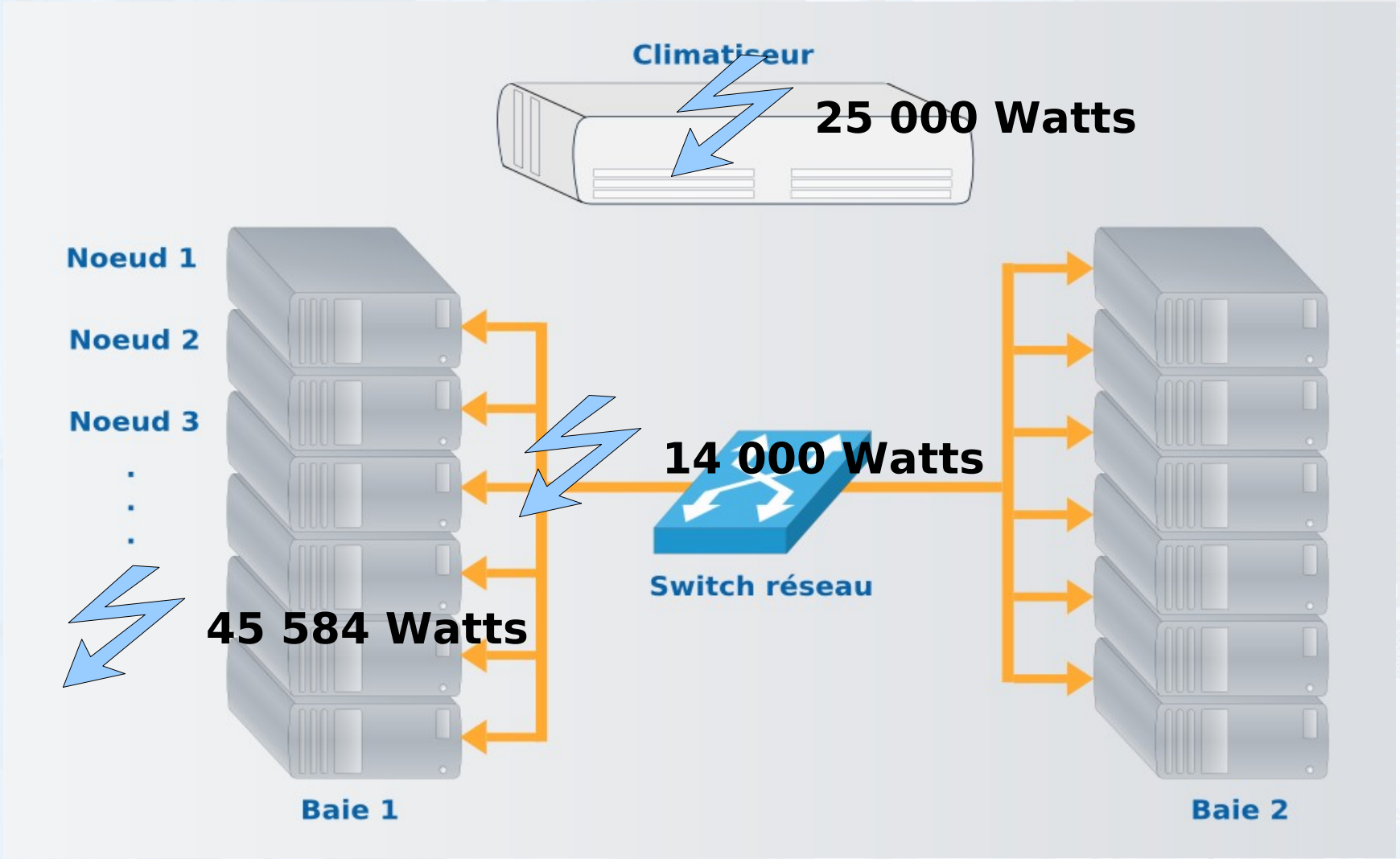
Configurable scheduler

Application check-pointing

Power saving in Kerrighed clusters

Context: consumption – active

258 nodes cluster



Total : 84 584 Watts

Context: consumption – inactive

Actual problem

100% of cluster usage: 84 584 Watts

0% of cluster usage : ~75 000 Watts

A cluster consumes the same amount of energy being used or not

Huge waste of energy in case of under-usage of a cluster

Impact on the energy bill

Impact on the environment

Goal of the project

Goal: limit the energy consumption

Idea: switch-off nodes unused or lowly used

Immediate saving on node consumption

Less heat: saving on air cooling system

Going a bit further

Adjust resource usage

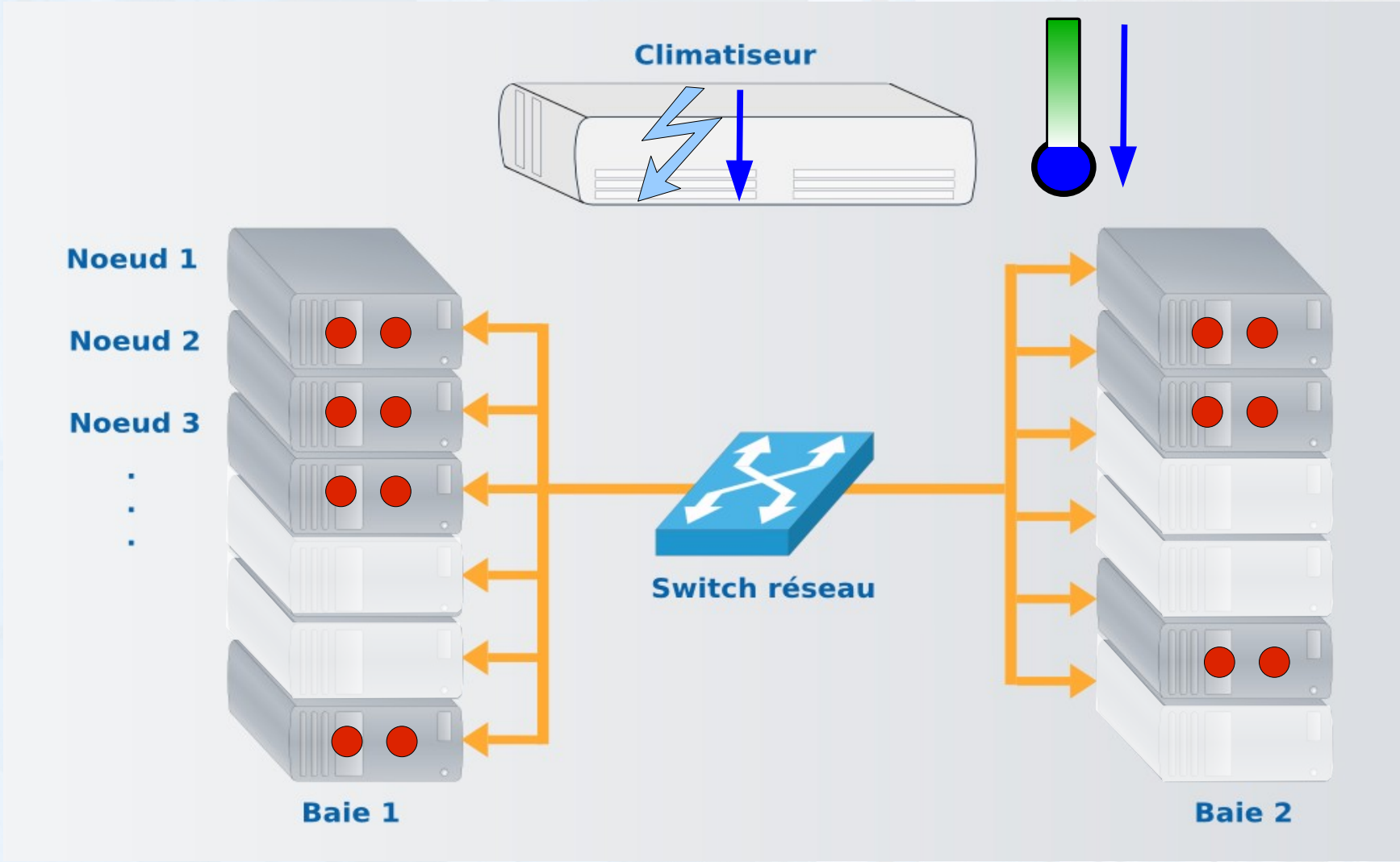
Processor frequency

Disc switch-on / switch-off

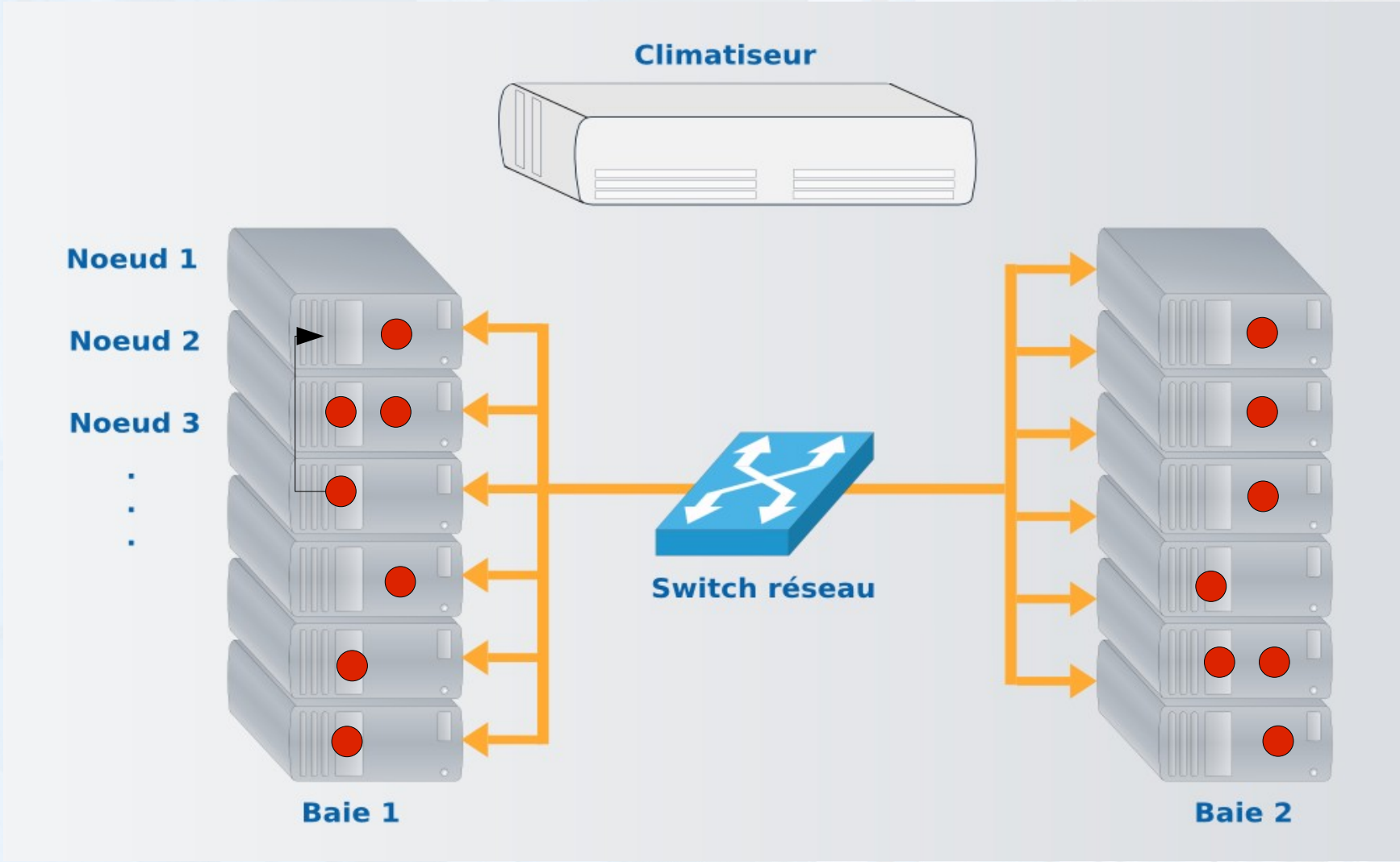
...

Limit hot area in the cluster by optimizing task placement

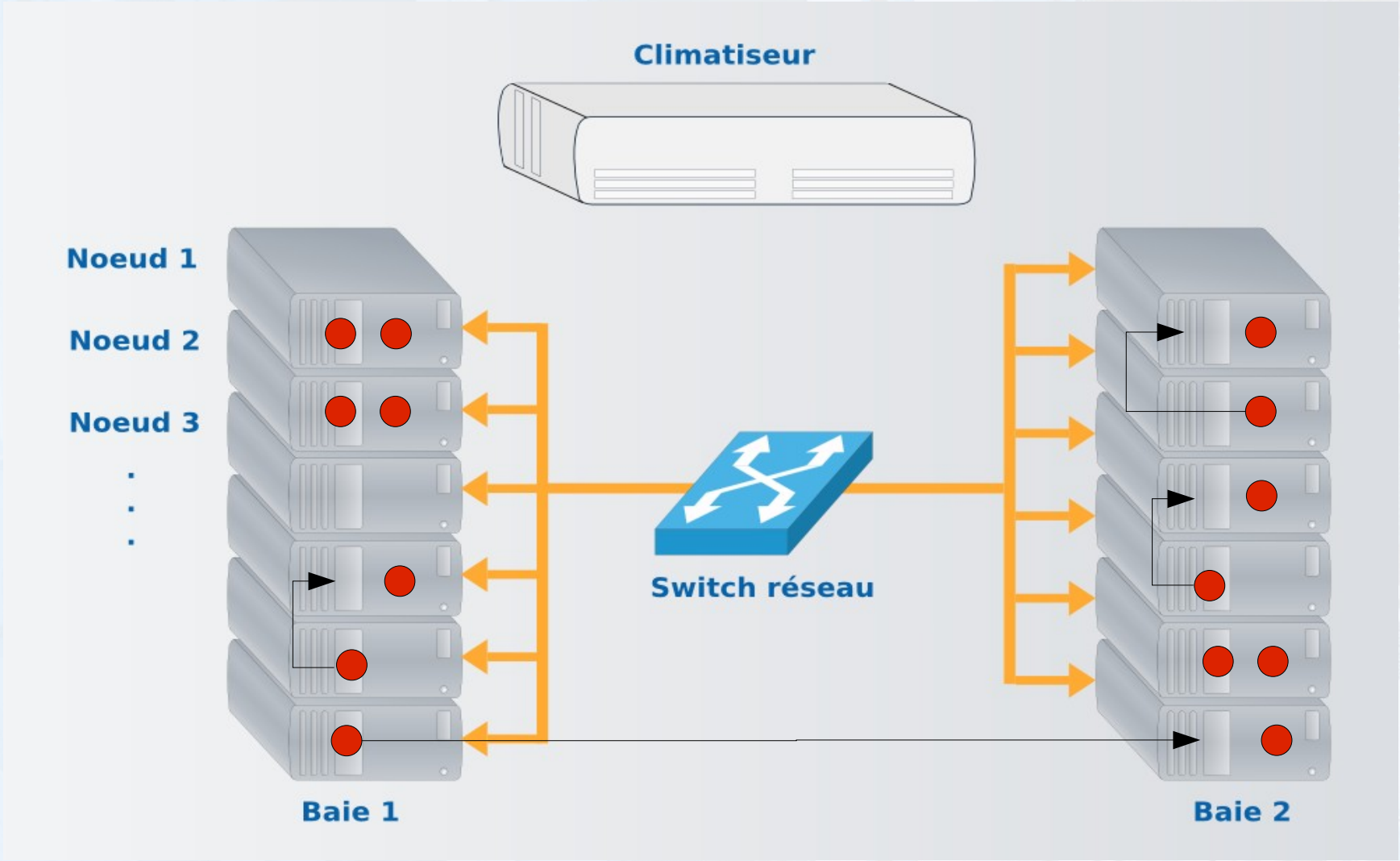
Technical issues: case 1



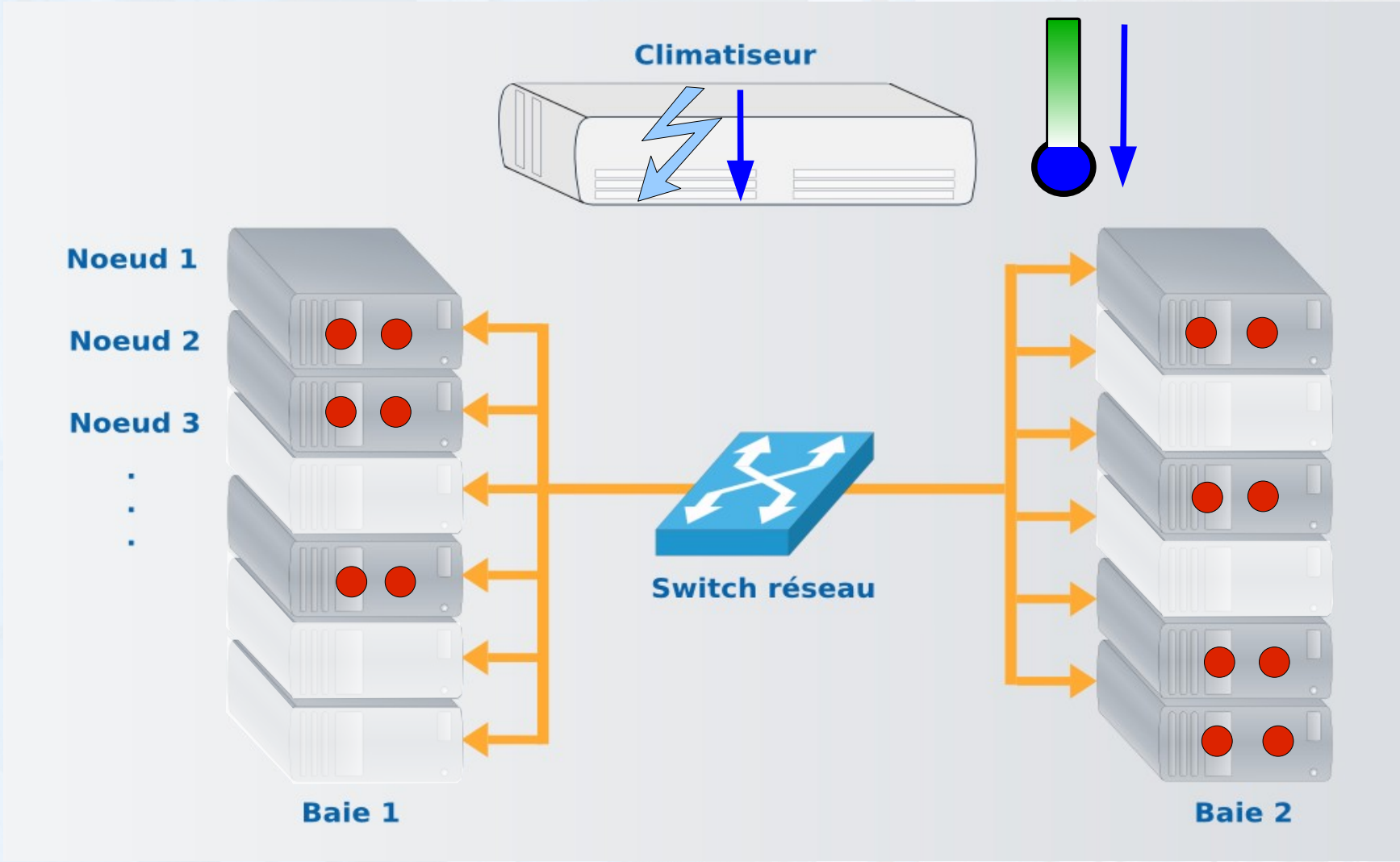
Technical issues: case 2



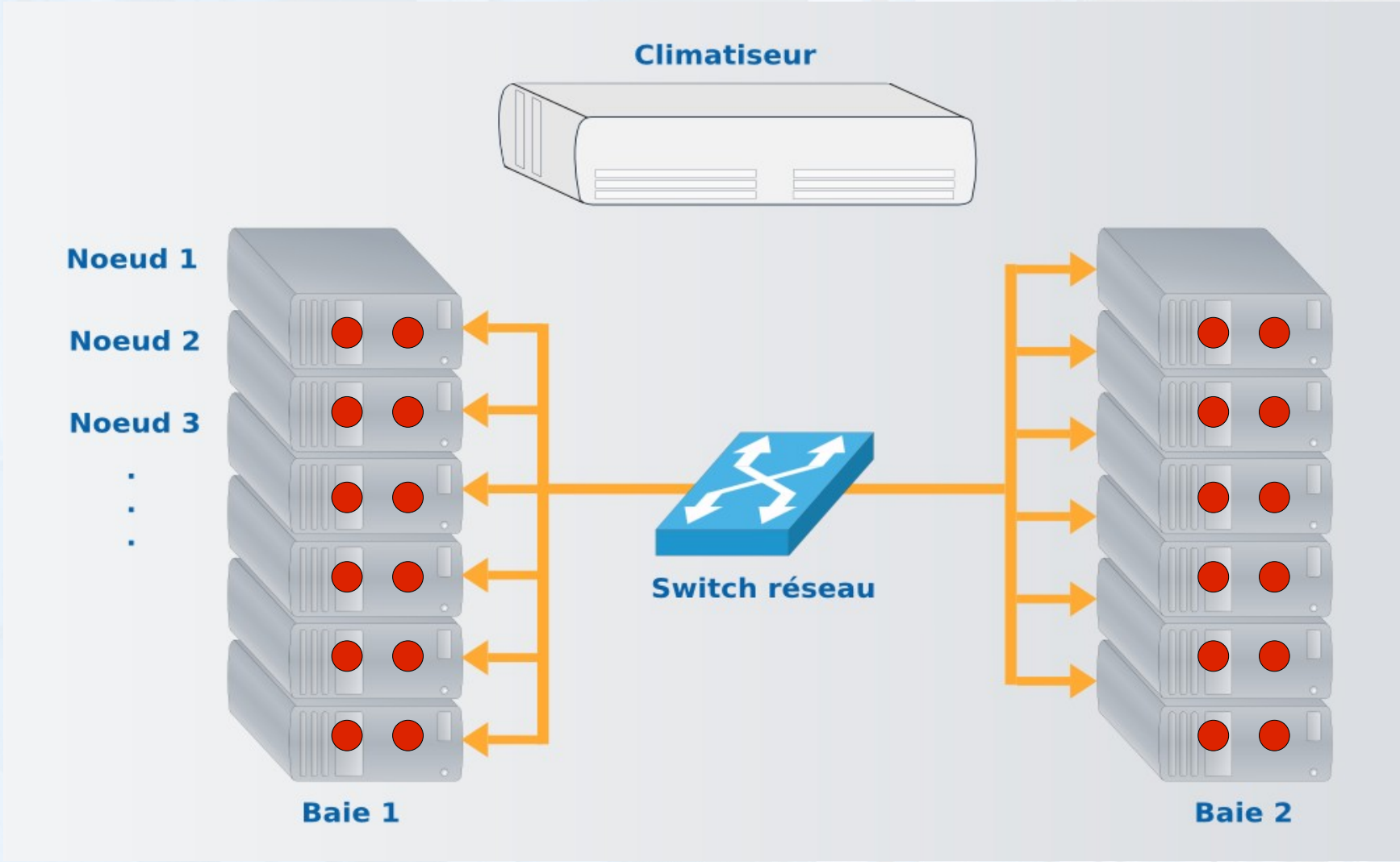
Technical issues: case 2



Technical issues: case 2



Technical issues: case 3



Perspectives

On going development

Port to linux 2.6.30

Node addition / removal

Re-enable dynamic streams

Distributed IP address

Integration of new DFS

2 on-going ANR

EcoGrappe

Power saving in Kerrighed clusters

Peta QCD

Scalability

Checkpoint of communicating applications