Analysing Manycore OS Design Aspects in NIX

Gorka Guardiola
paurea@lsub.org
LSUB
Rey Juan Carlos Univ.

Noah Evans
Jim McKie
Bell Labs

Ron Minnich
Google

Charles Forsyth
Vita Nuova

vita nuova

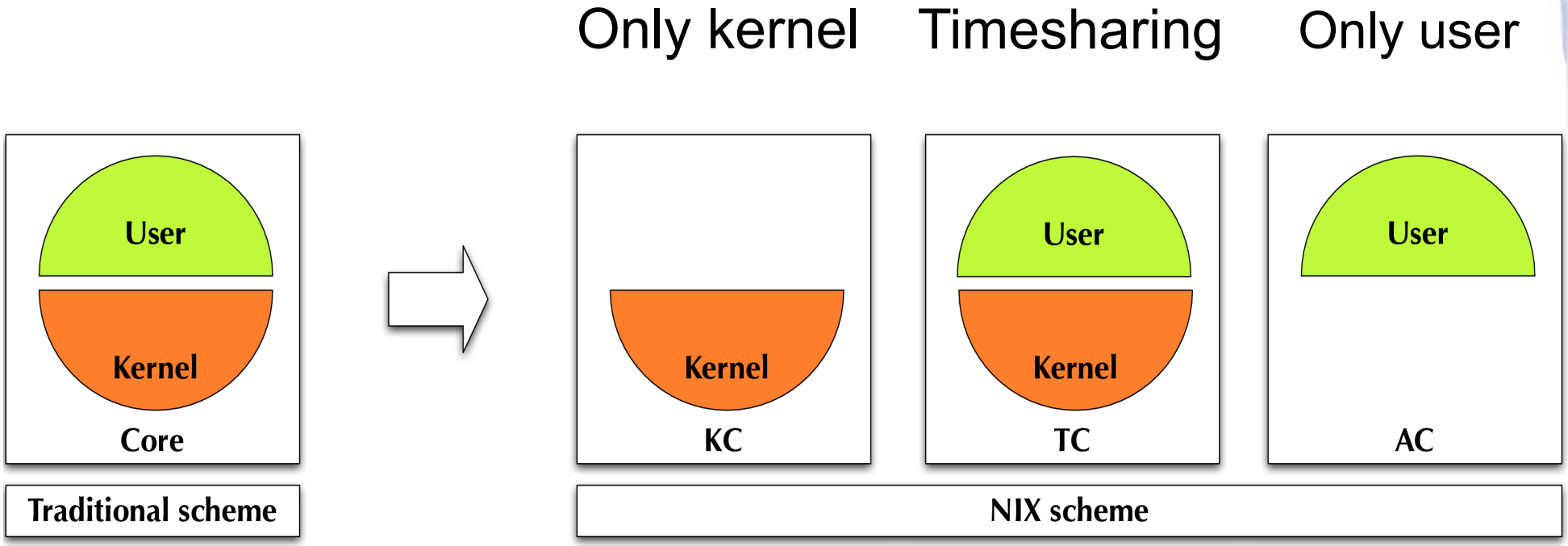
Fco. J. Ballesteros
Enrique Soriano
Rey Juan Carlos Univ.

What is NIX?

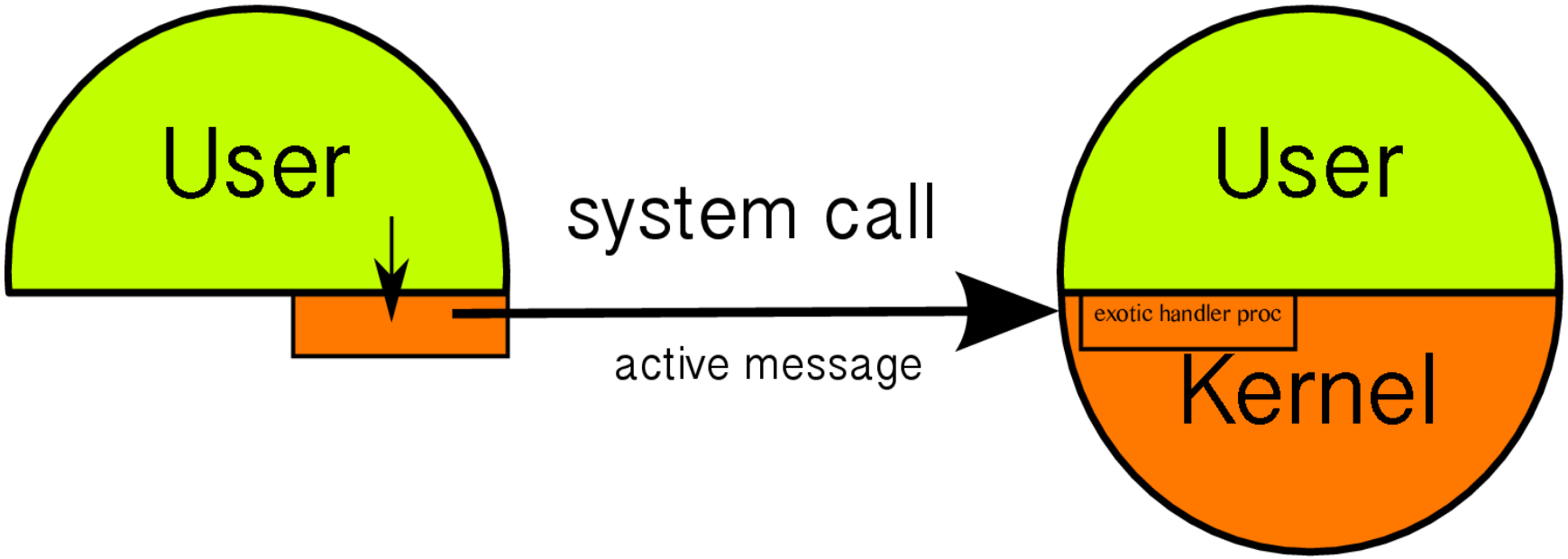
- Joint work, Bell Labs, Rey Juan Carlos Univ., Vitanuova, Sandia Nat. Labs
- Based on Plan 9
- Rethink OS abstractions and assumptions for new multicore machines
- Try to keep the kernel in charge of policies

Roles

Kernel space/user space **core** specialization



Roles



AC core

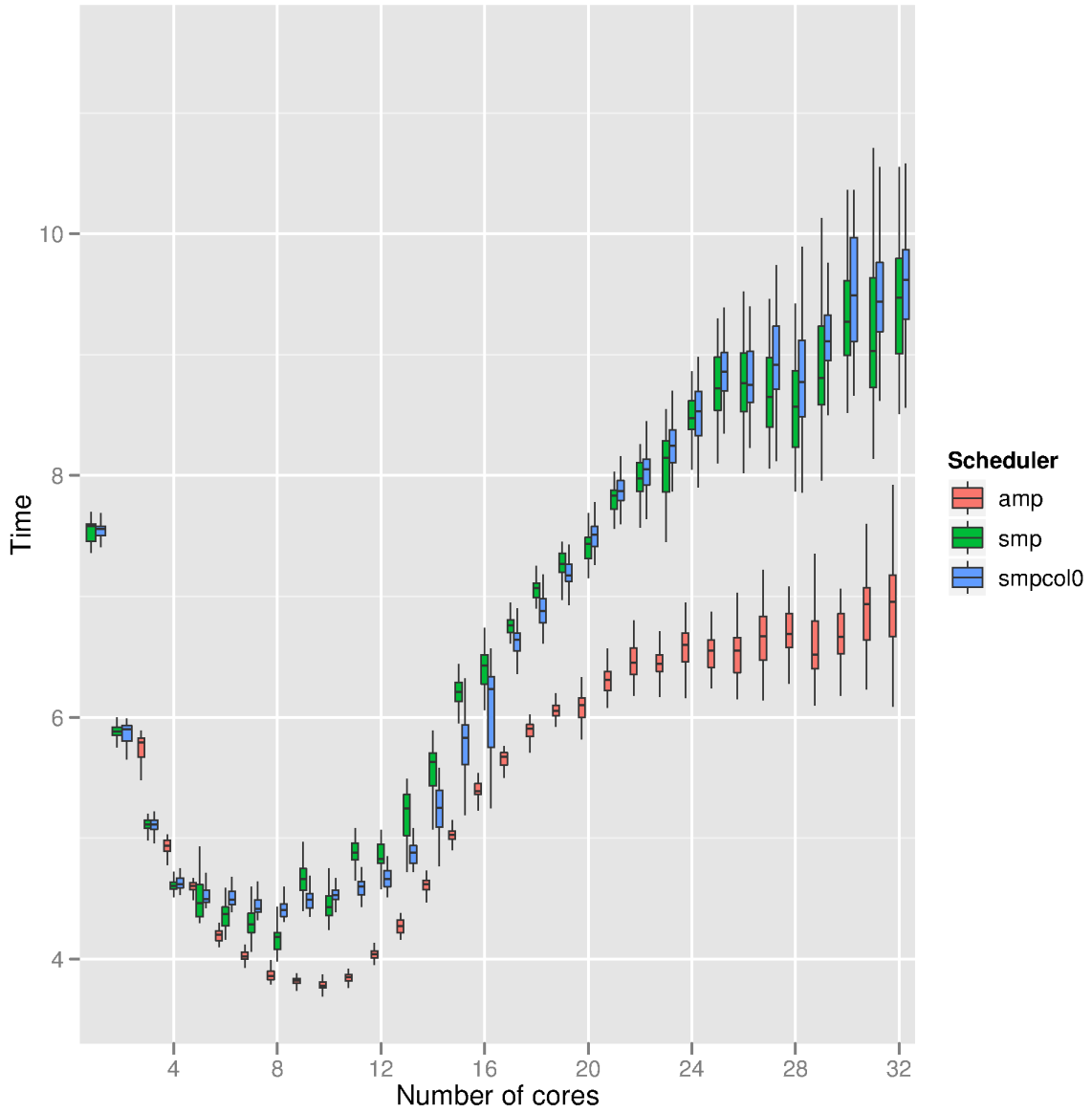
TC core

Kernel space is in a core with kernel

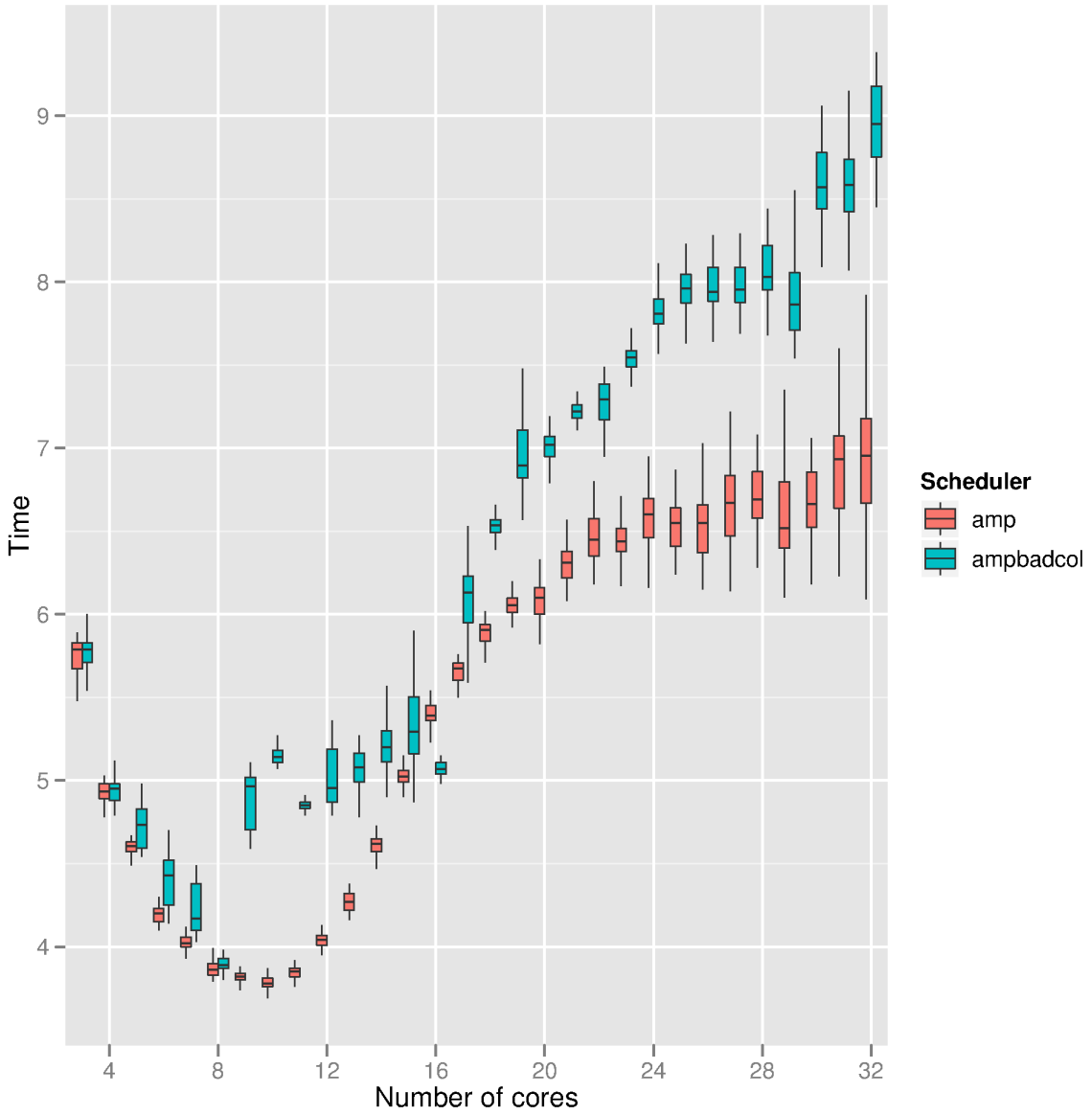
Roles

- Mechanisms: **may not have coherent memory** (the implementation does for now)
- Processors may be specialized (hw)
- Assign roles **automatically**, wip
- New roles (XC), wip
- Automatic assignment, policies, wip

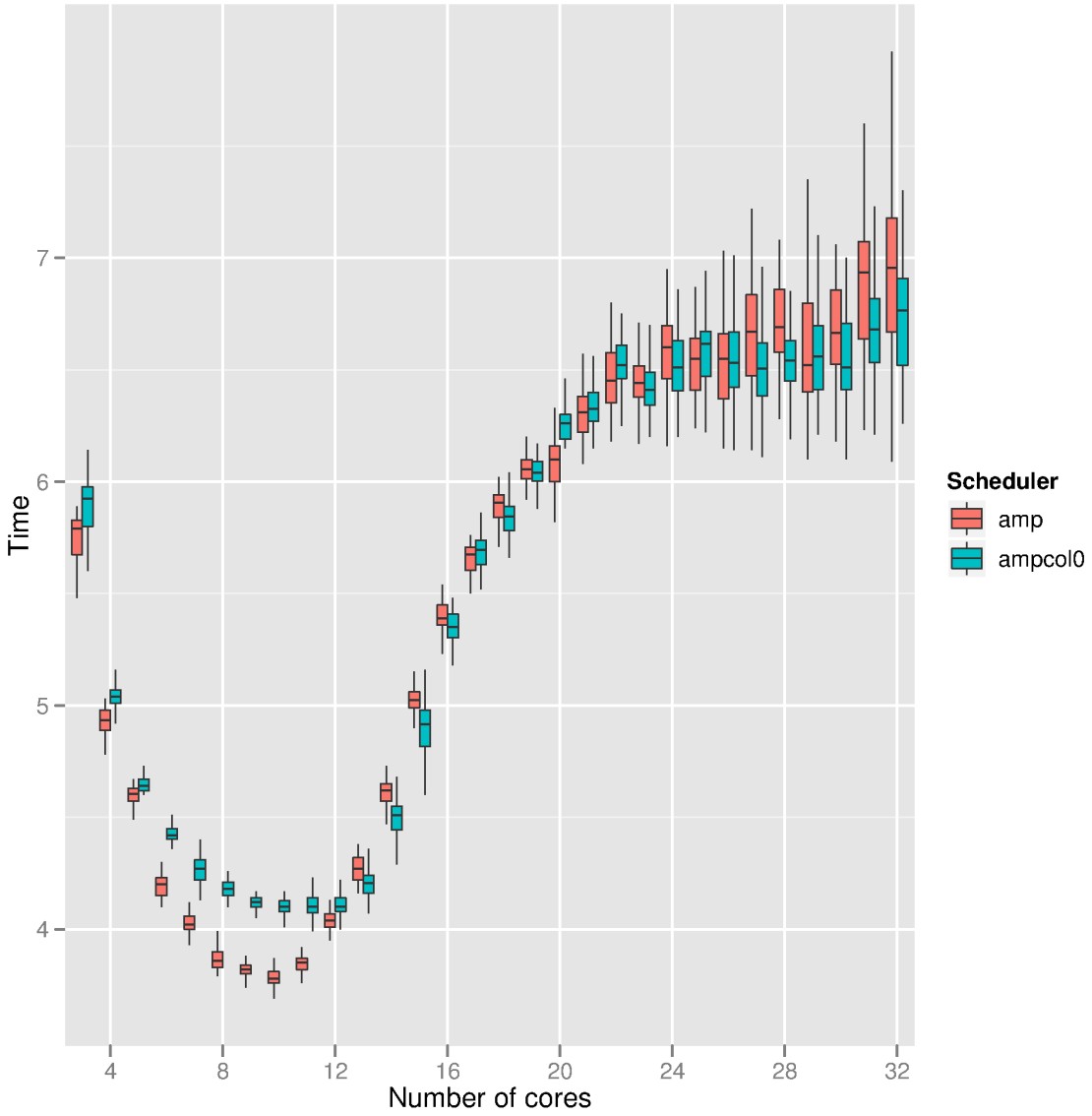
Scheduler


Time
10
8
6
4
4
8
12
16
20
24
28
32
Number of cores
Scheduler
amp
smp
smpcol0

Scheduler


Time
9
8
7
6
5
4
4
8
12
16
20
24
28
32
Number of cores
Scheduler
amp
ampbadcol

Scheduler


Time
7
6
5
4
4
8
12
16
20
24
28
32
Number of cores
Scheduler
amp
ampcol0

Scheduler

- Found that adding cores made compiling a kernel slower (K10 magny cours, 32 cores)
- AMP vs SMP
- NUMA aware
- Latency aware? (not in ACPI info)
- Probably the next bottleneck is in spinlocks, find which, segregate the resources when possible, wip

No questions :-)

- http://lsub.org/ls/nix.html
- http://code.google.com/p/nix-os

I have some questions myself

- I have some, latency vs. bandwidth vs. CPU bound
- ACPI information on latency
- How do different architectures compare