OSPERT 2013

9th annual workshop on

Operating Systems Platforms for Embedded Real-Time applications

July 9, 2013. Paris, France

held in conjunction with

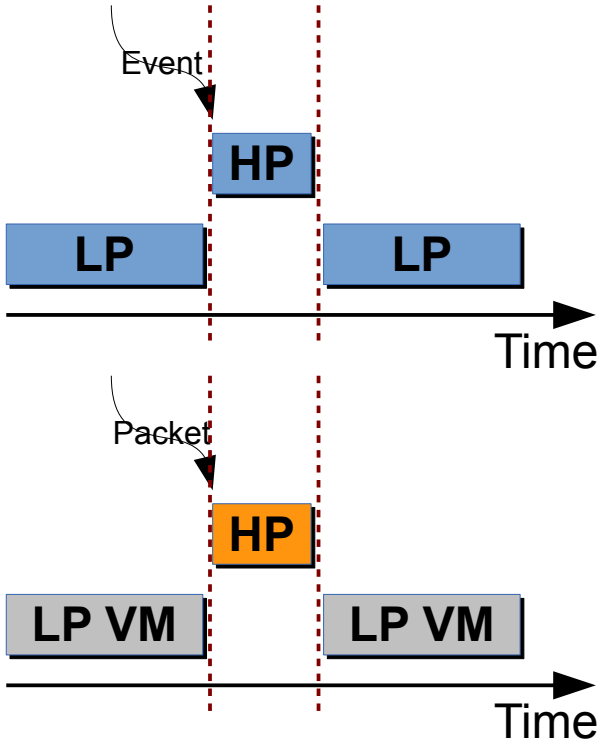
Priority Inheritance on Condition Variables

Tommaso Cucinotta
Bell Laboratories, Alcatel-Lucent
Dublin, Ireland

Introduction

In computing systems, it is convenient to have **tasks with different priorities**

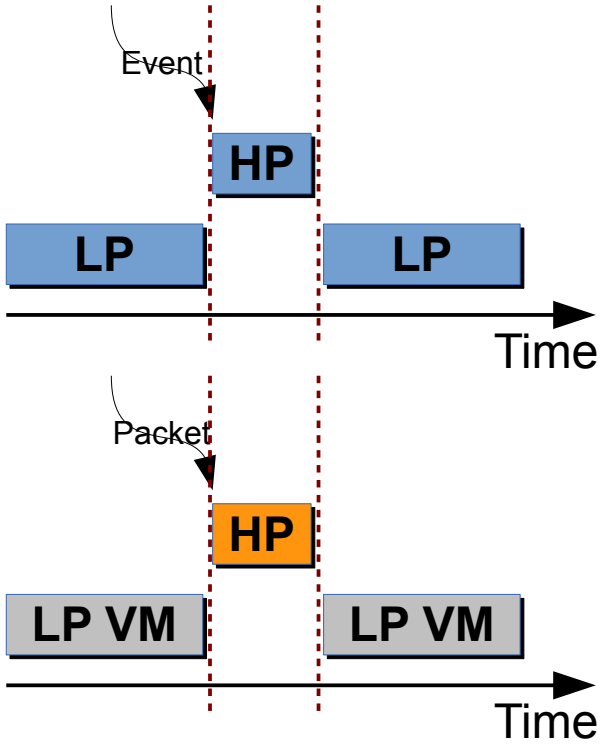
- to ensure low-latency and responsiveness
- I/O prioritized over computing
- Virtual Machines in cloud infrastructures run at different priority/urgency level
  - **gold** vs. **bronze** customers



Introduction

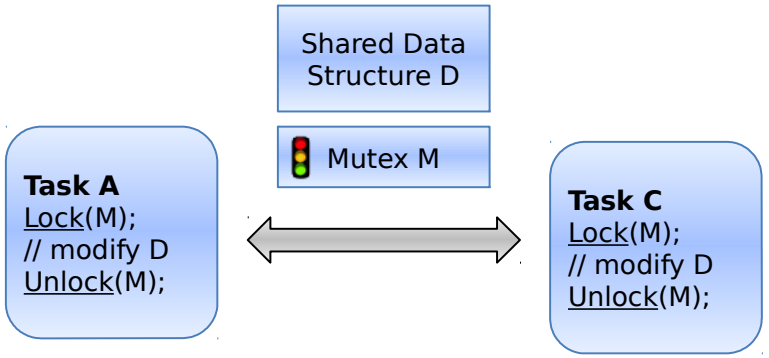
In computing systems, it is convenient to have **tasks with different priorities**

- to ensure low-latency and responsiveness
- I/O prioritized over computing
- Virtual Machines in cloud infrastructures run at different priority/urgency level
  - **gold** vs. **bronze** customers



Interactions among tasks often

- use shared data structures in memory
- **serialize access** through a **mutual exclusion semaphore** (mutex)



Introduction

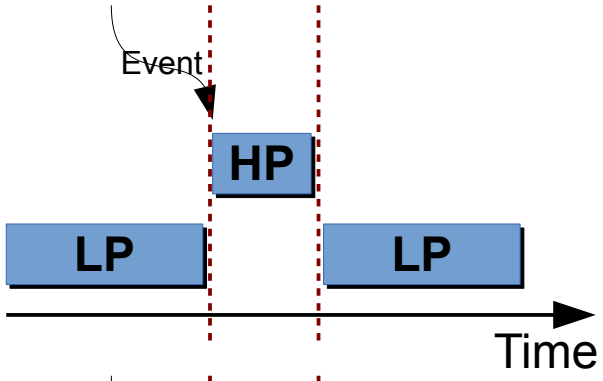
In computing systems, it is convenient to have **tasks with different priorities**

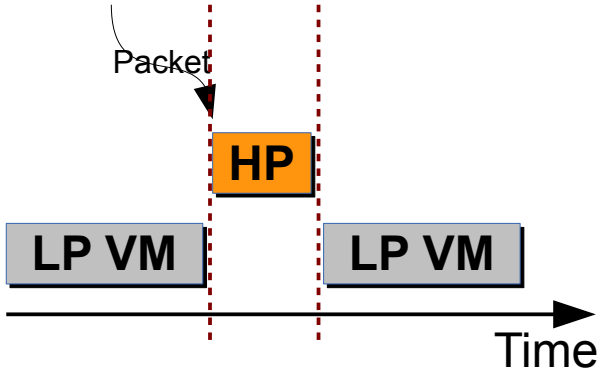
- to ensure low-latency and responsiveness
- I/O prioritized over computing
- Virtual Machines in cloud infrastructures run at different priority/urgency level
  - **gold** vs. **bronze** customers

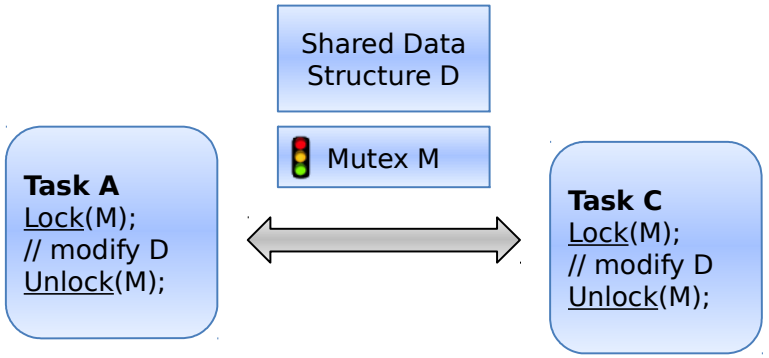
Interactions among tasks often

- use shared data structures in memory
- **serialize access** through a **mutual exclusion semaphore** (mutex)

Mixing these two paradigms leads to **undesirable situations**



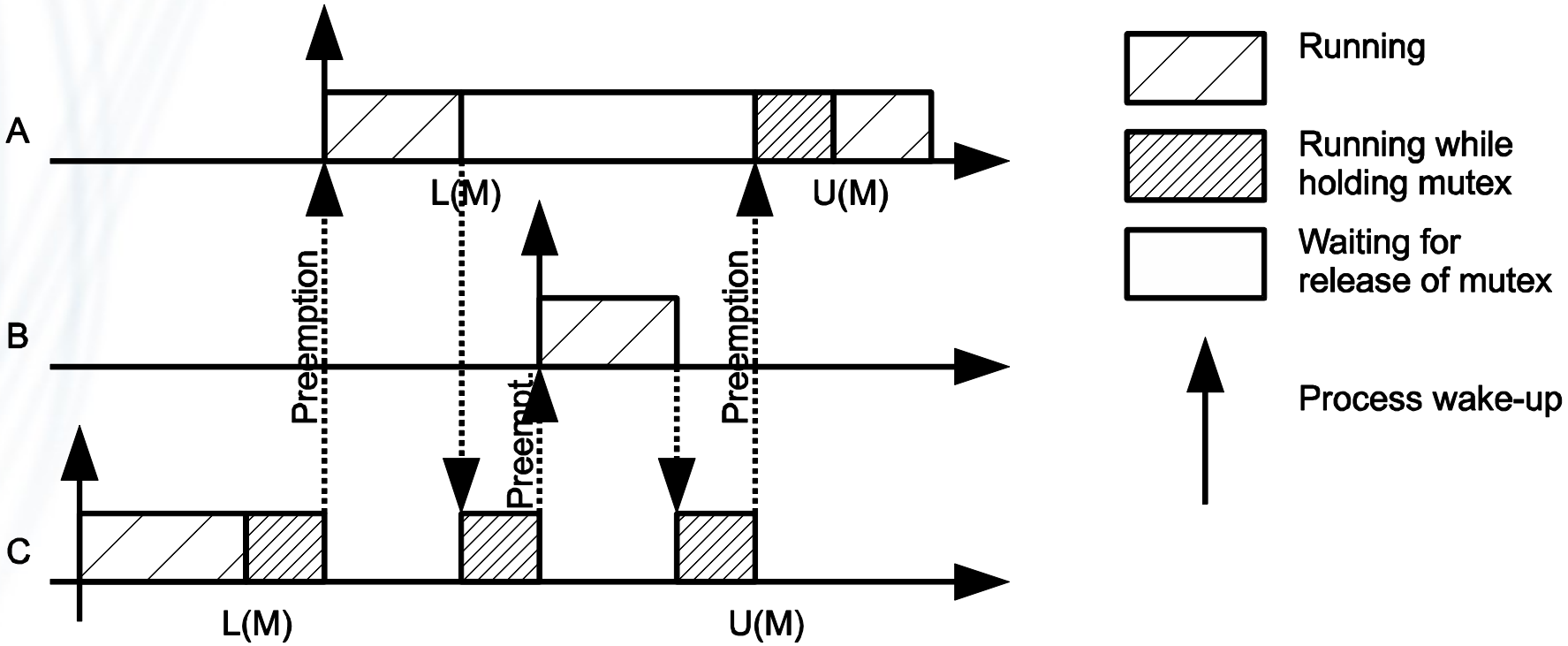




Problem of Priority Inversion on Mutexes

Priority Inversion occurs when

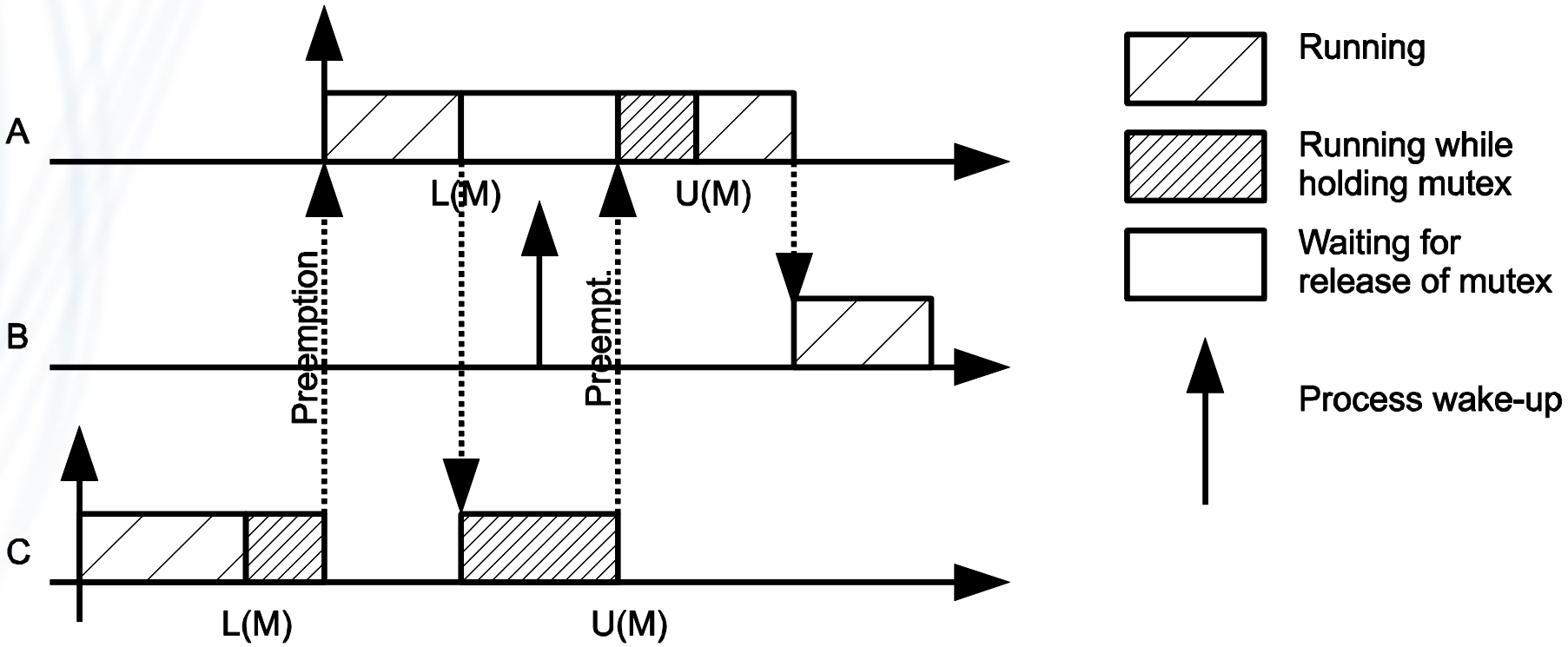
- HP task A synchronizes with LP task C
- ... but a middle-priority task B defers execution of C, therefore A



Fixing Priority Inversion on Mutexes

Priority Inheritance avoids the problem

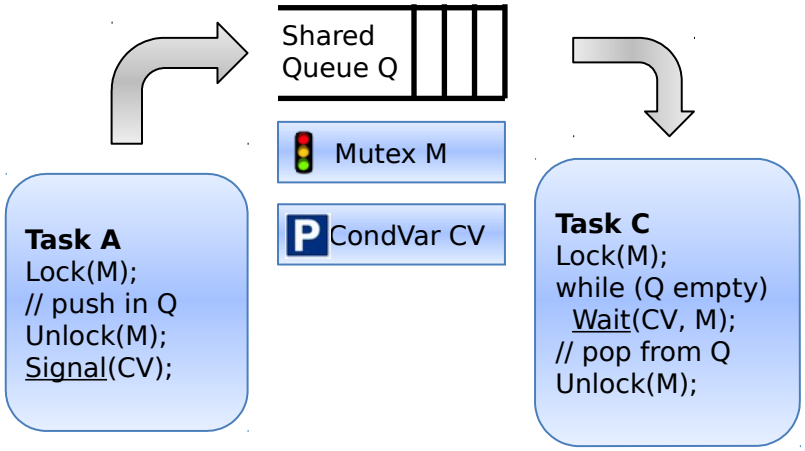
- mutex owner **inherits** highest priority among tasks waiting for mutex unlock (if higher than its own priority)



More Problems

More **complex interactions** require

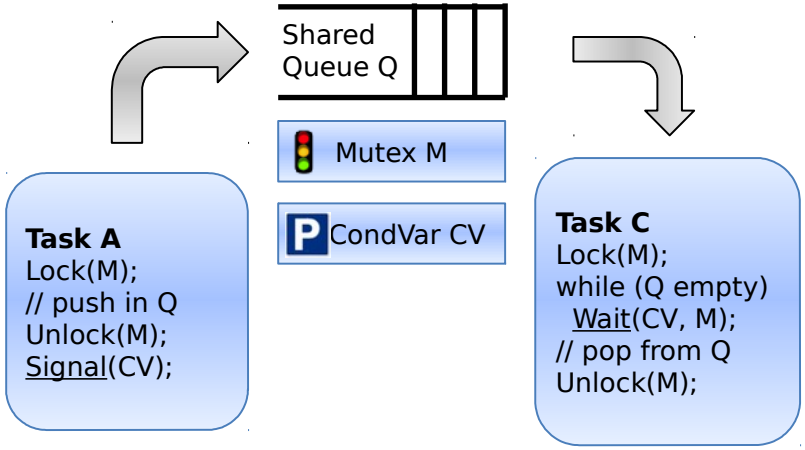
- a **condition variable** (condvar)
  - over which tasks may suspend waiting for a condition, before the critical section



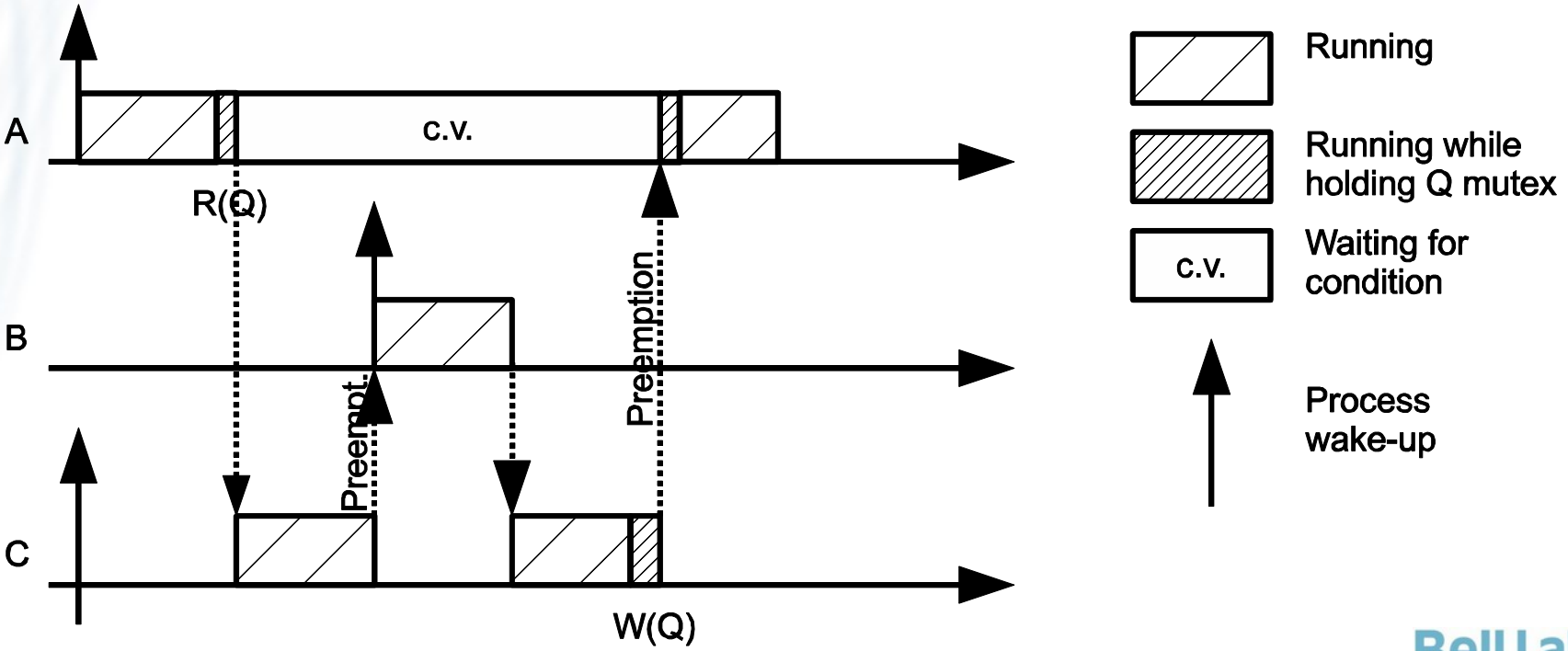
Background Information

More **complex interactions** require

- a **condition variable** (condvar)
  - over which tasks may suspend waiting for a condition, before the critical section



Priority Inversion still occurs when



Related Work

Cornhill & Sha, '87 (International Workshop on Real-Time Ada Issues)

- Indefinite delay of HP tasks by LP tasks

Sha et al., '90

- Basic Priority Inheritance and Priority Ceiling Protocol

Later

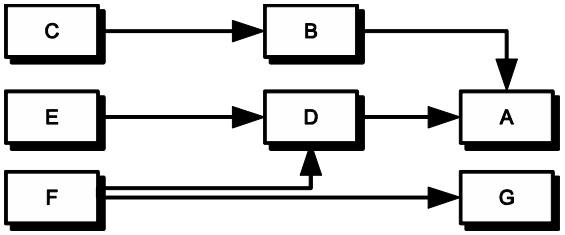
- SRP, BWI, MBWI, FMLP, others...

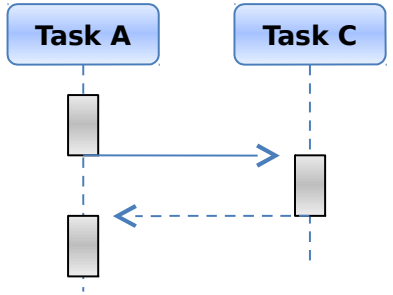
Proposal: novel **general solution**

- for the problem of priority inversion
- in presence of arbitrary interactions among tasks
- based on mutexes and condition variables

Problem **only marginally** addressed in RT literature

- Limited to blocking RPC/RMI/Client-Server case
  (condition helper implicitly known)

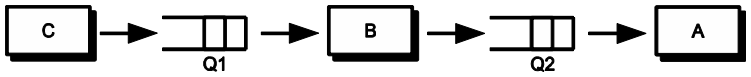


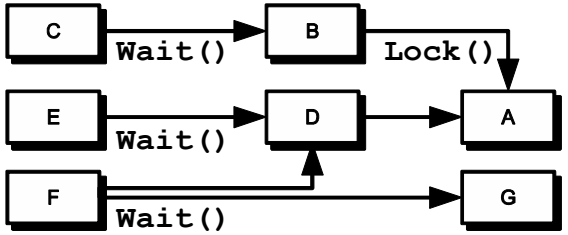


Proposed Solution: PI-CV

Priority Inheritance on Condition Variable (PI-CV)

- **declare which tasks may signal()** on a condition variable (the helpers)
- helpers automatically inherit highest priority among wait()-ers
  - (if higher than their own priority)
- inheritance cancelled on signal()
- **transitive behaviour**
  - C inherits from B, which inherits from A
  - integrate with classical priority inheritance mutexes





How to realize it ?

- add **new syscall**, e.g., to POSIX pthreads
  - `pthread_cond_helpers_add(condvar, thread)`
  - `pthread_cond_helpers_del(condvar, thread)`
- **kernel modifications**
  - (futex code on Linux)

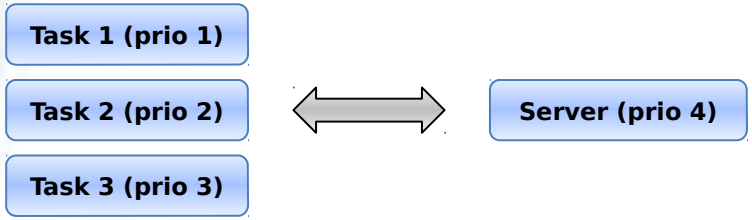
Simulation

PI-CV implemented in RTSim

- open-source simulator from SSSA

Simulated scenario

- Client-server interactions (see table)

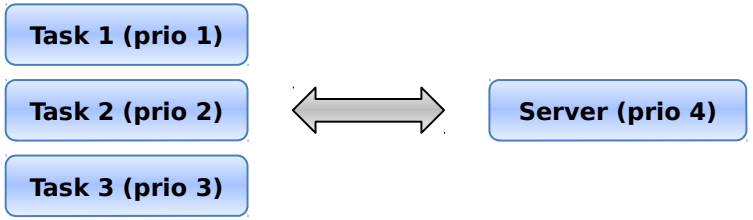


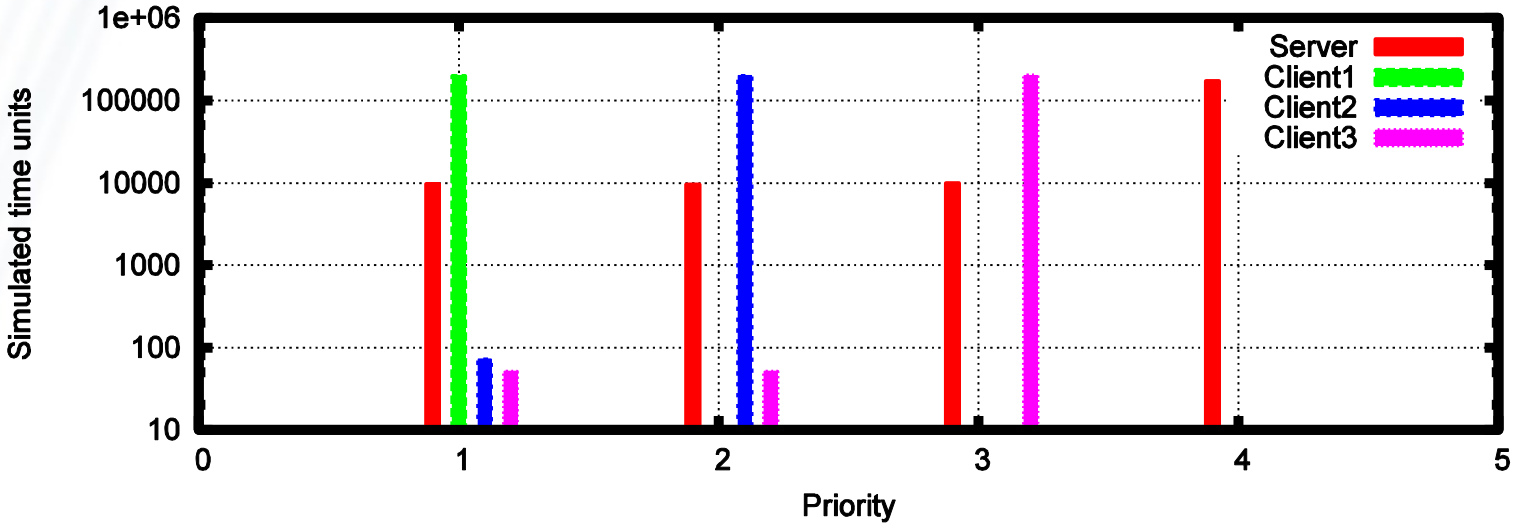
Parameter	Client1	Client2	Client3
Task period	676	683	687
Ovh lock/unlock	1	1	1
Ovh wait/signal	2	2	2
Ovh push/pop	2	2	2
Job comp	50	50	50
Server call	20	20	20
Exp duration	200000		

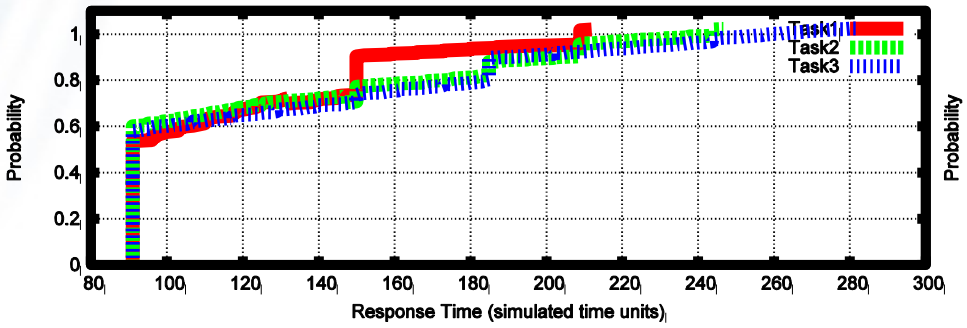
Simulation Results

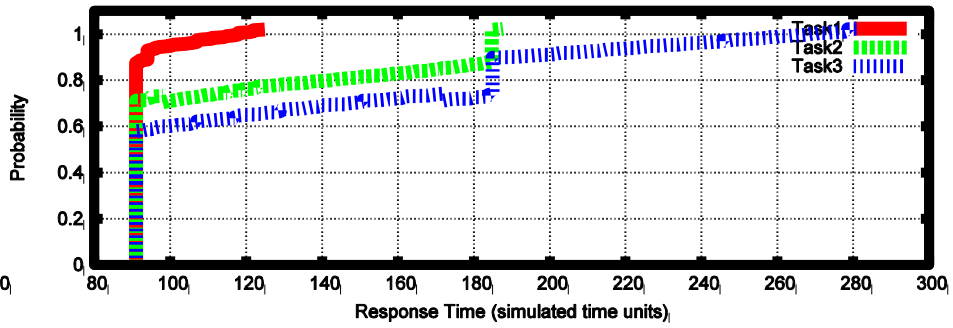
Results

- **41% reduction of WCET** for HP task (as due to avoiding priority inversion)







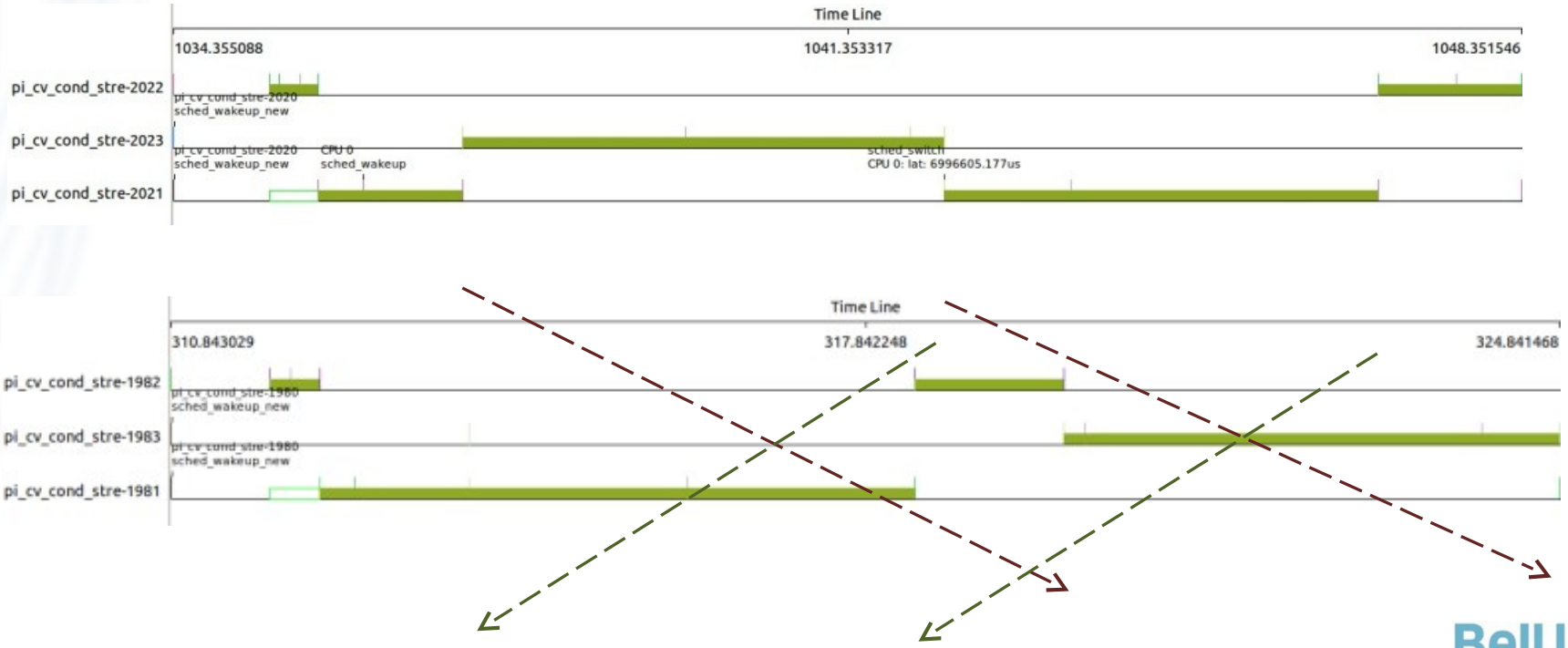


Current Status & Future Work

Future Work

- **Real implementation** on Linux (half-way)
- **Use-case study with real application**
- **Schedulability analysis** (theoretical)

On-going collaboration with Scuola Superiore Sant'Anna and University of Trento



Q&A

Thanks for your attention!

Questions?