Linux com a sistema operatiu de temps real

Treball de final de carrera

Enginyeria Informàtica
Gener del 2013

Francesc Guim Bernat (consultor)
Carles Marsal Castillero (alumne)

Departament d'Arquitectura de Computadors i Sistemes Operatius.
Universitat Oberta de Catalunya

Motivació

- Progressiva adaptació de Linux com a S.O. de sistemes encastats.
- Plataforma robusta, escalable, adaptable, estandarditzada i lliure.
- Suport de gran quantitat d'aplicacions i ampli recolzament per part de fabricants.
- Ara bé, **és Linux capaç de satisfer els requeriments de temps real habituals en aquest tipus de sistemes?**

Objectius

- **Determinar i avaluar les configuracions alternatives de Linux encarades al temps real.**

  1. Que és el temps real?
  2. Components del sistema operatiu involucrats.
  3. Millores introduïdes pel seu assoliment.
  4. Avaluació de les diferents alternatives.

Abast

- Línia principal del nucli de Linux, sèrie 2.6/3
- Sistema uniprocessador (UP)
- Plataforma x86
  - Intel Atom N270 @ 1,6Ghz
  - 1GB RAM
  - 32GB SSD
- Queden excloses alternatives de nucli dual (Xenomai, RTAI...)

Planificació

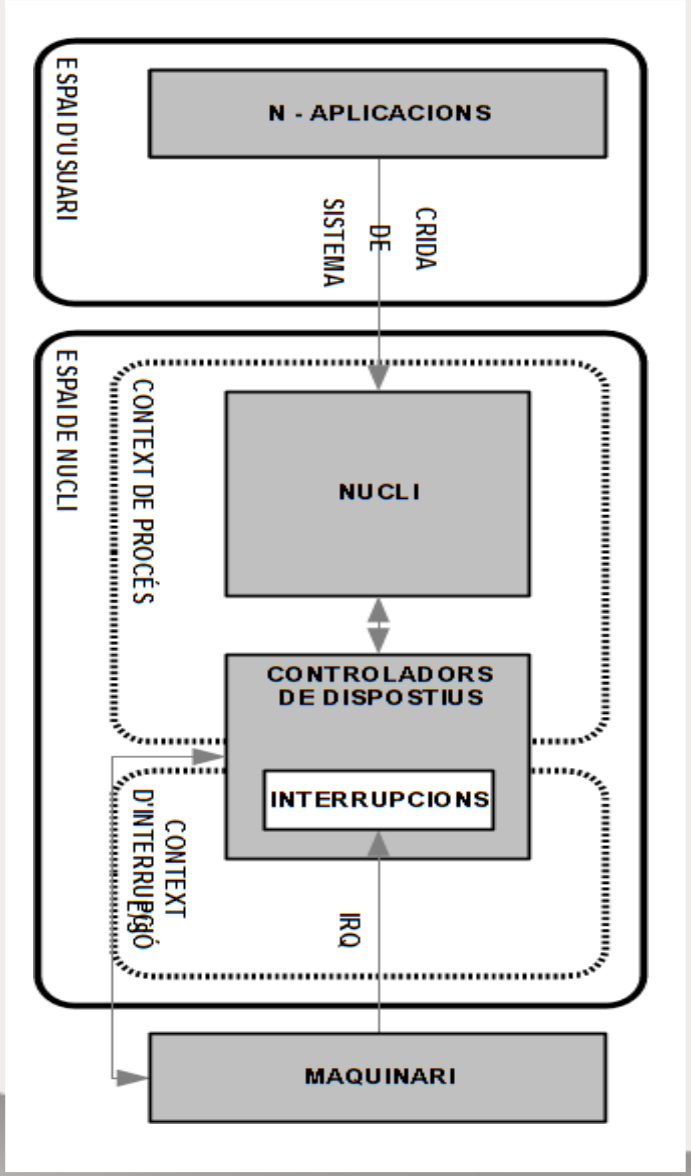
Tasca	Prevista	Real	Desviació
Cerca d'informació	10/10/12	19/10/12	+34d
Exposició base teòrica	15/10/12	19/11/12	
Configuració, compilació i prova de nuclis	08/11/12	03/11/12	-5d
Selecció i prova d'eines de *benchmarking*	18/11/12	30/12/12	+42d
Disseny i prova entorn d'estrès	18/11/12	30/12/12	
Execució de les proves	28/11/12	07/01/13	+39d
Implementació exemple aplicació RT	03/12/12	-	-
Anàlisi dels resultats	08/12/12	11/01/13	+33d
Elaboració de la memòria	15/12/12	20/01/13	+35d
Elaboració de la presentació	22/12/12	22/01/13	+30d

* Els càlculs de desviació agrupats responen a tasques desenvolupades de forma iterativa i conjunta. Es mostra la desviació acumulada.

Definició de temps real

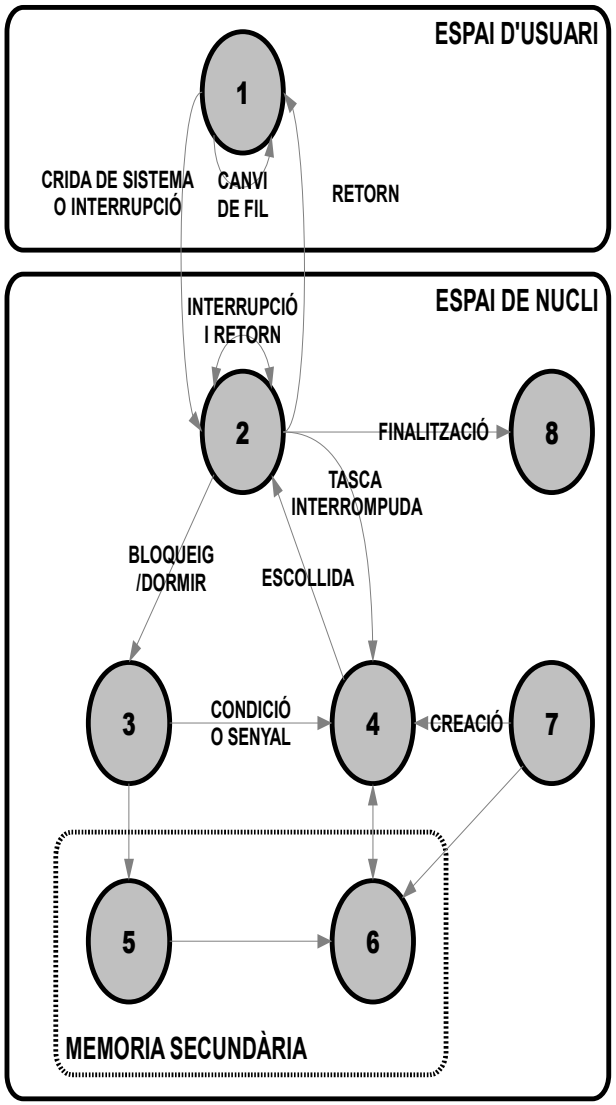
- **Compliment de terminis temporals fixats.**
- Condicions:
  - **Latència** = temps de resposta del sistema.
  - **Determinisme** = predictible, *jitter* baix.
- Segons les conseqüències de no compliment:
  - Hard real time: invalida sistema.
  - Firm real time: invalida resposta.
  - Soft real time: devalua resposta.

Arquitectura



- **Espai d'usuari:**
  - Execució d'aplicacions.
  - Accés a maquinari via crides.
- **Espai de nucli:**
  - Privilegis accés maquinari
  - Execució de:
    - Peticions espai d'usuari
    - Interrupcions de maquinari

Gestió de processos



- **Instància de programa**

1. Procés en execució
2. Realitzant una crida de sistema
3. En espera d'esdeveniment
4. Apunt per executar-se

5 i 6. 3 i 4 en memòria secundaria

7 i 8. Creació i finalització

Planificador

- SCHED_OTHER (Prioritat dinàmica, CFS)
  - Fins fi *timeslice* o tasca de prioritat fixa.
  - *Timeslice* dinàmic, CPU-bound vs IO-bound.
- **SCHED_FIFO (Prioritat estàtica, temps real)**
  - Per una mateixa prioritat política FIFO.
- **SCHED_RR (Prioritat estàtica, temps real)**
  - Per una mateixa prioritat política RR.
  - *Timeslice* estàtic.

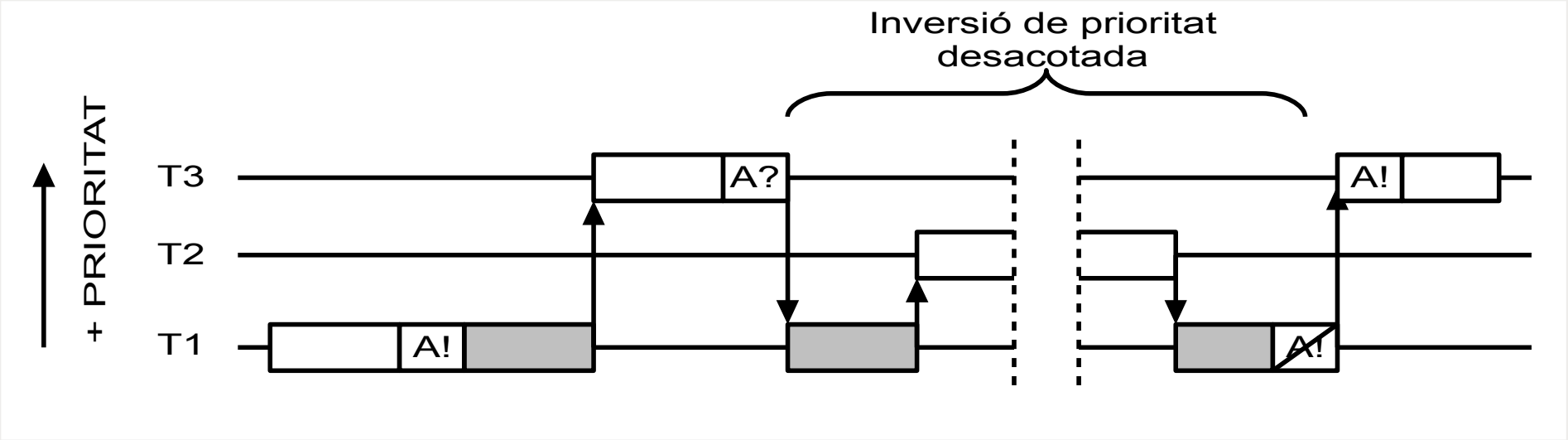
Interrupcions

- **Part alta (operacions crítiques)**:
  - Copia de la informació dels *buffers*.
  - Restabliment de la interrupció.
  - Execució instantània.
- **Part baixa (operacions no crítiques):**
  - Tractament de la informació / accions.
    - Al finalitzar part alta (softirq).
    - Diferida, en context de procés (ksoftirqd).

Sincronització

- Evita les condicions de carrera:
  - Interrupció vs nucli
  - Processador vs processador (SMP)
  - Concurrència nucli (nucli preemptiu)
- Mecanismes, entre altres:
  - **Spin-locks (SMP):** bucle, no bloqueig tasca!
  - **Semàfors/*mutex*:** bloqueig tasca (*sleep*).
  - **Preempt_disable:** protecció concurrència nucli.

Inversió de prioritats

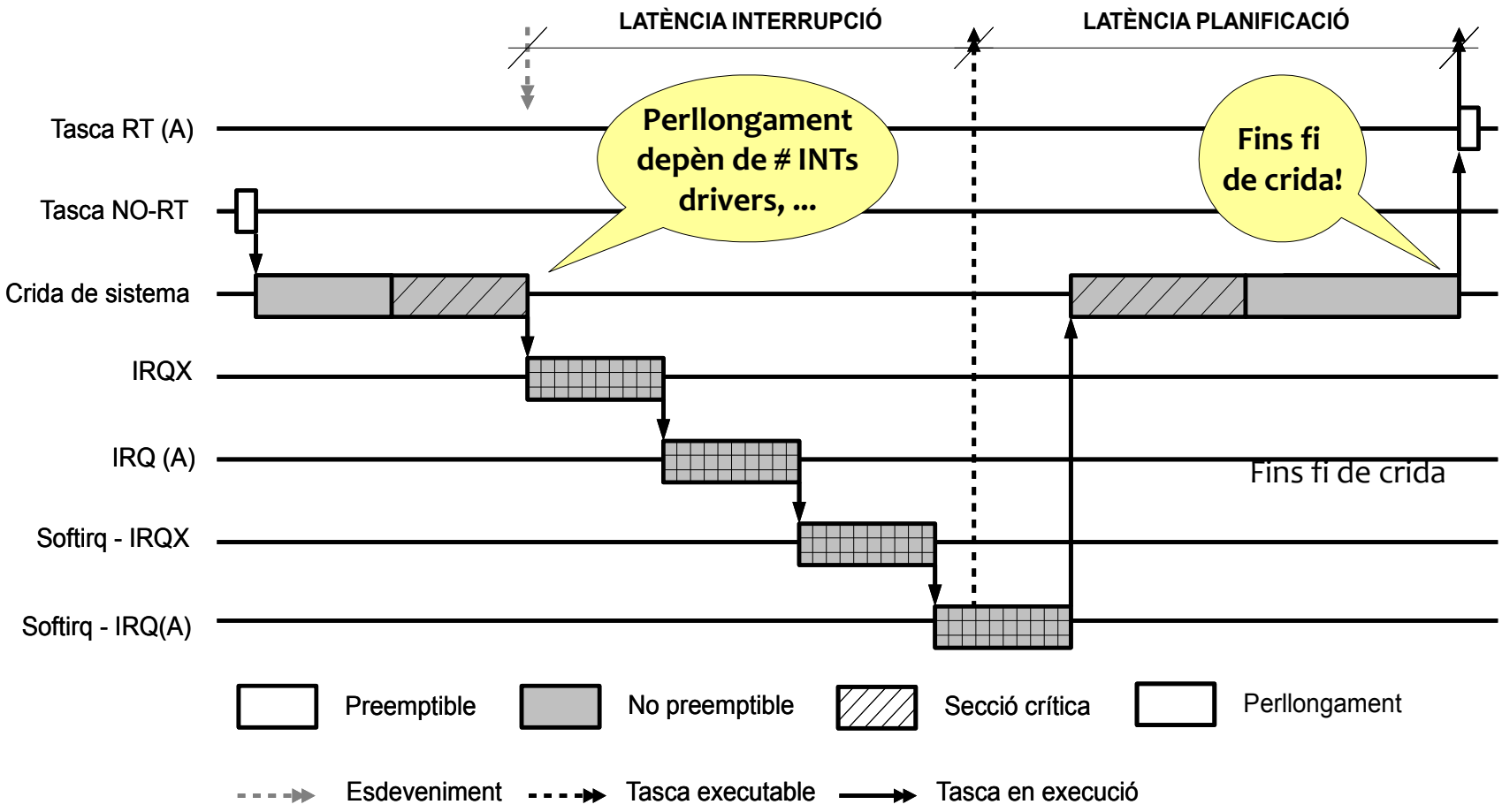


- Tasca de major prioritat (T3) necessita A.
- T1 < T3 s'ha d'executar per alliberar A.
- T2 < T3 interromp T1 < T2
- **Inversió desacotada (T2<T3), quan acabarà T2 ?**

Modalitat NO PREEMPT

- **Nucli no preemptible**
- Crides no s'interrompen
- Longitud de la crides:
  - >100ms en alguns casos!
  - Pitjor cas difícil de determinar
    - Longitud de llistes, múltiples camins...
  - Indeterminada
    - Pot ser interrompuda (INTs) i estesa

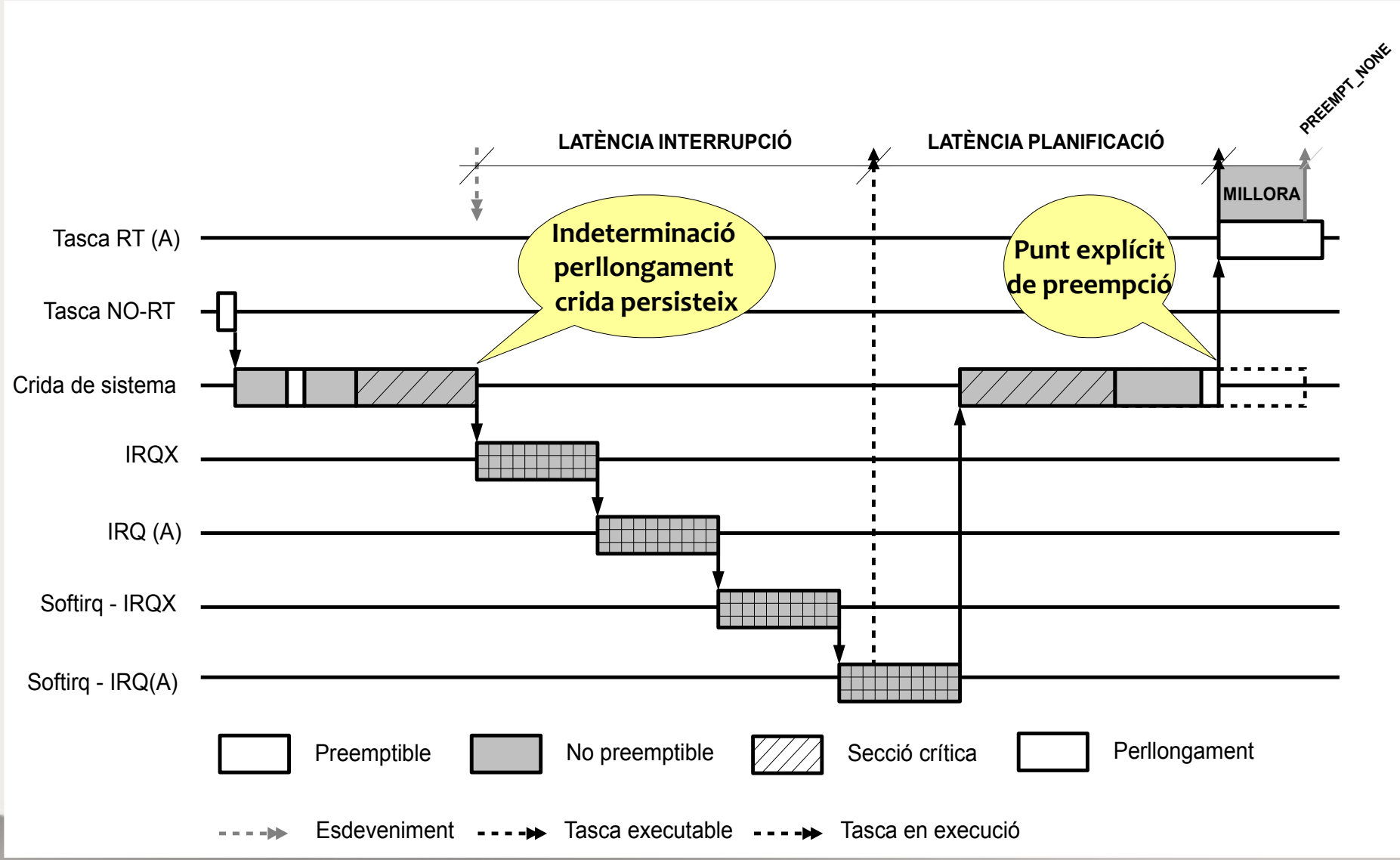
Modalitat NO PREEMPT



Modalitat VOLUNTARY

- **Punts de preempció implícits**
- Longitud de les crides “es redueix”:
  - Pitjor cas
    - Determinat experimentalment
    - Introducció de punts de preempció
    - Iteracions llistes *(lock-breaking)*
  - Indeterminada, es manté dificultat RT
    - Pot ser interrompuda (INTs) i estesa

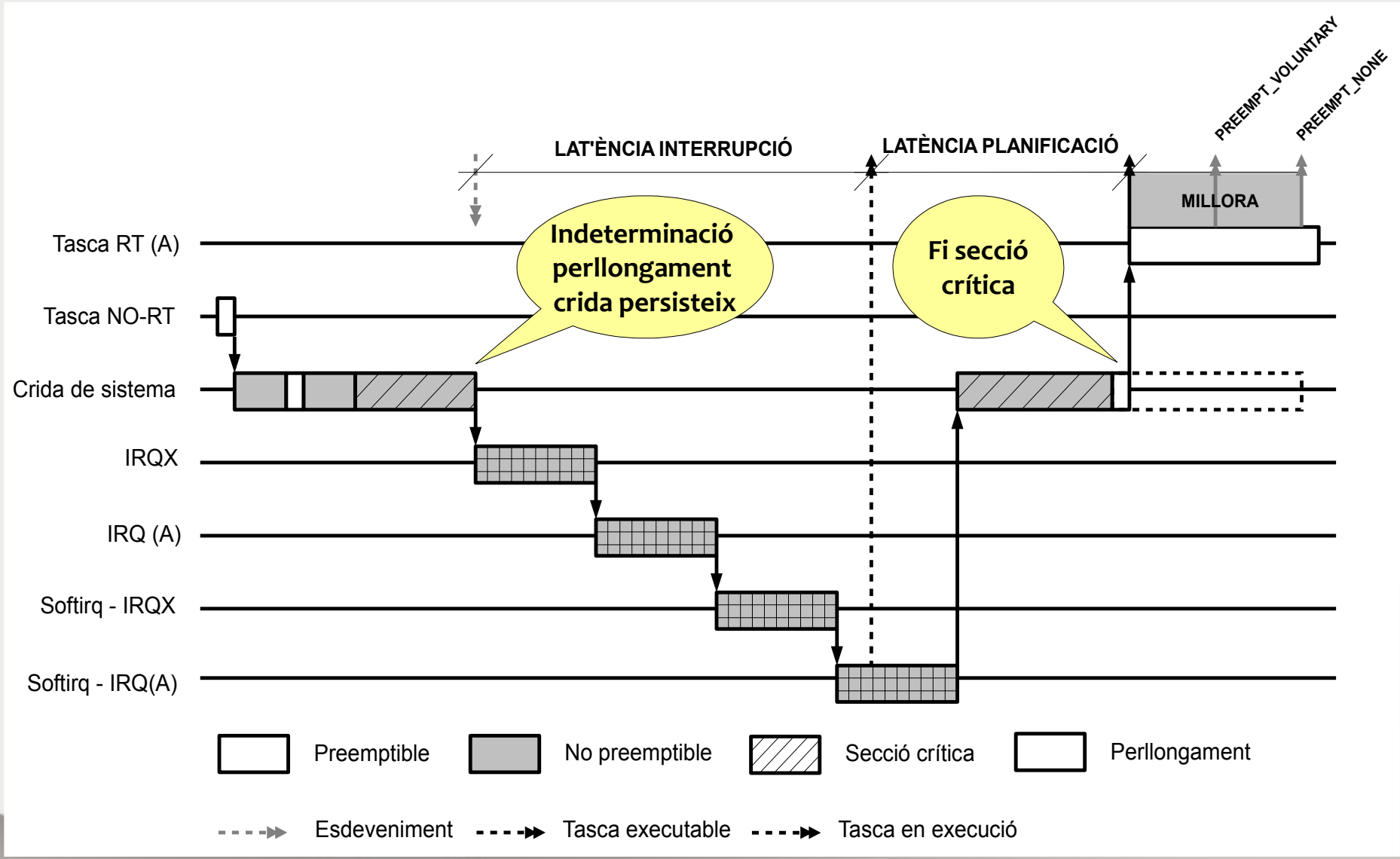
Modalitat VOLUNTARY



Modalitat PREEMPT

- **Nucli preemptible**
- Excepte seccions crítiques
- Longitud limitada a seccions crítiques
  - Pitjor cas
    - Depèn seccions crítiques
    - Es manté *lock-breaking*
  - Indeterminada, es manté dificultat RT
    - Pot ser interrompuda (INTs) i extesa

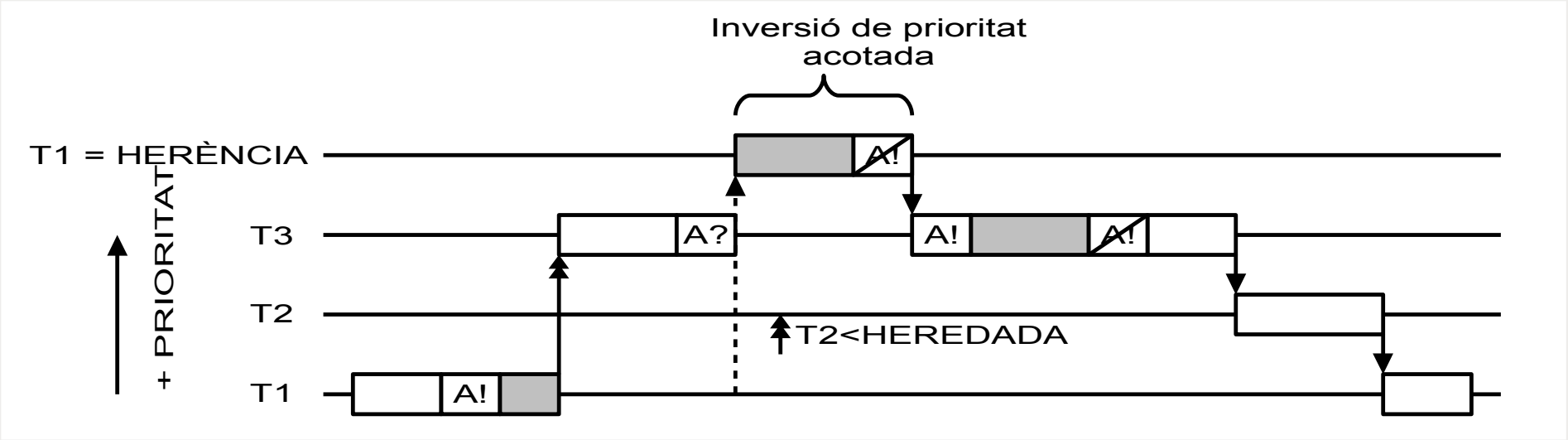
Modalitat PREEMPT



Thread IRQS / PI-Mutex (PATCH)

- Aportacions PATCH RT
  - Thread IRQS
    - Interrupcions en context de procés
    - Possibilitat establir prioritats
    - Possibilitat de tasca > gestió interrupció
  - *PI Mutex*
    - *Mutex* (semàfors binaris)
    - Herència de prioritats (PI)

Herència de prioritats (PATCH)

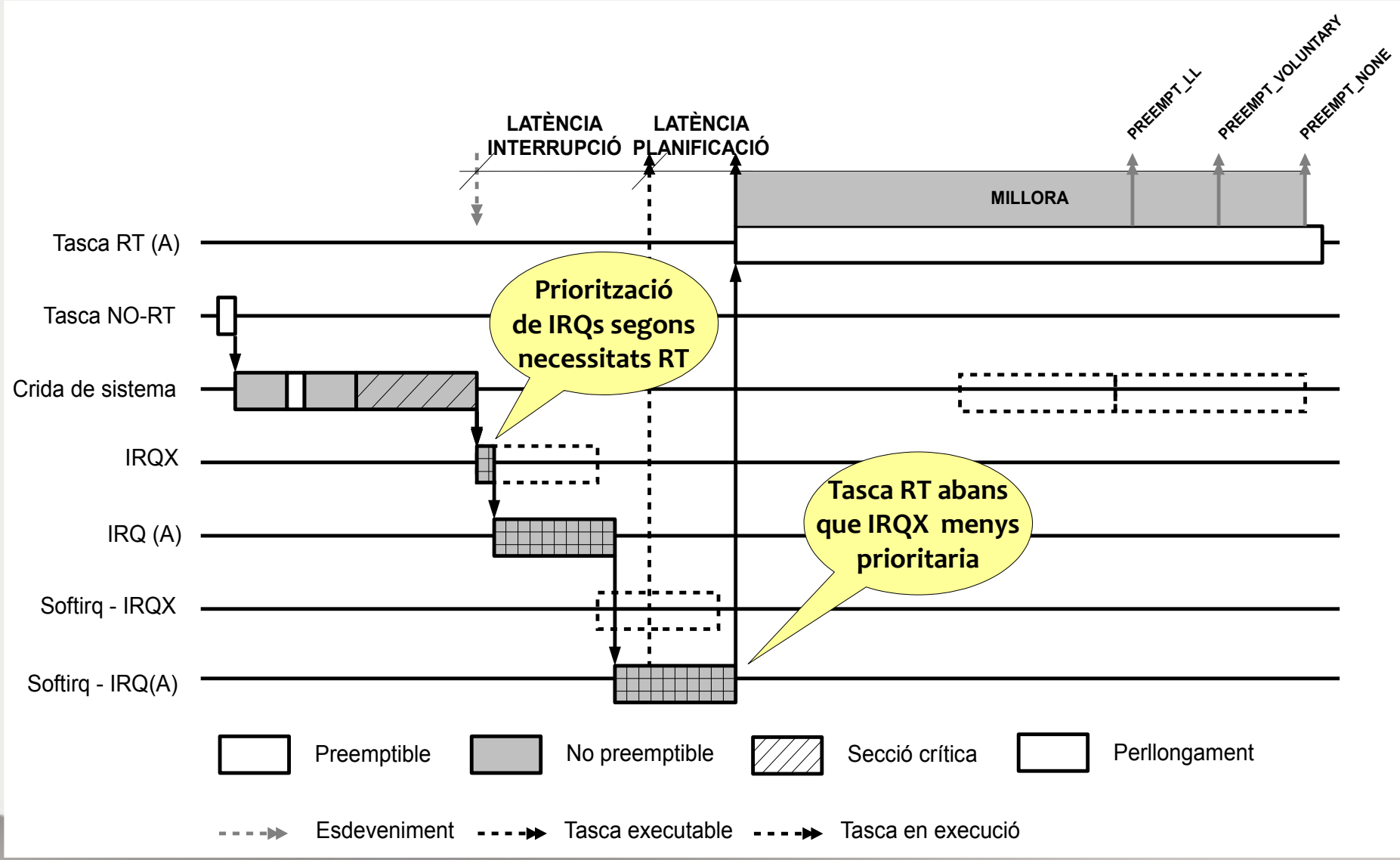


- Tasca de major prioritat (T3) necessita A.
- T1 < T3 s'ha d'executar per alliberar A.
- T2 < T3 no interromp T1 que ha heretat T3 > T2
- **Inversió ACOTADA (T1<T3), fins fi secció crítica**

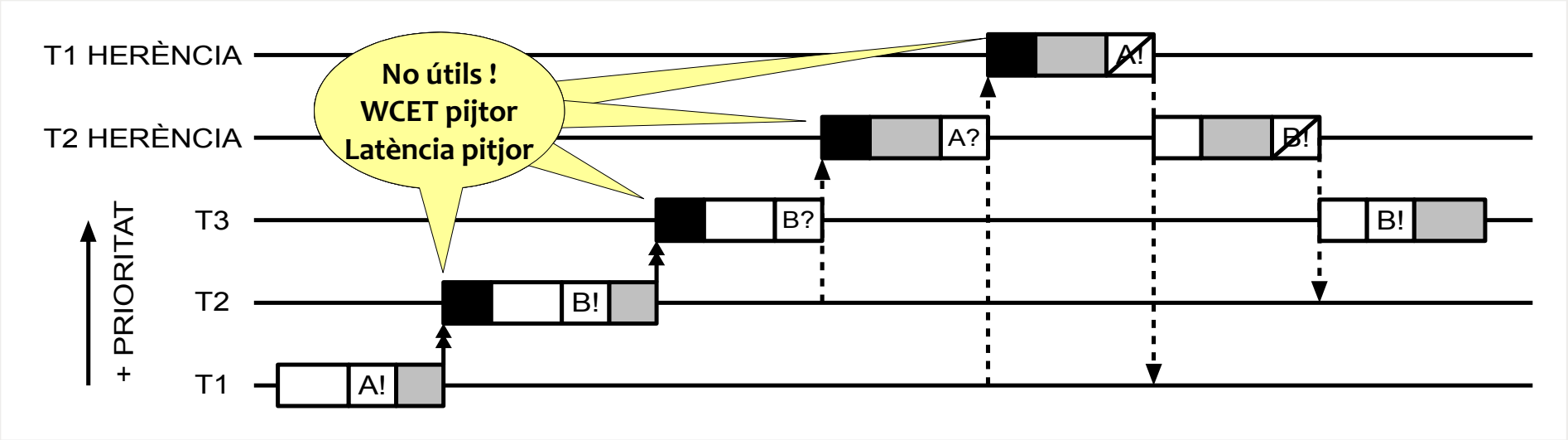
Modalitat PREEMPT_RT (PATCH)

- **Nucli TOTALMENT preemptible (99,999%)**
- Incloses seccions crítiques
  - Ús de PI Mutex
  - Cost preempció seccions crítiques!
- Thread IRQs
  - Establiment de prioritats
  - Tasques RT per sobre d'interrupcions
- T (interrupció + planificació) $\longrightarrow \varnothing$

Modalitat PREEMPT_RT (PATCH)



Cost preempció seccions crítiques



- T2 interromp T1 secció crítica A
- T3 interromp T2 secció crítica B , ...
- T3 necessita B, T2 necessita A, ... hereten
- **+ 4 canvis de context no útils!**

Tècniques de programació

- Polítiques RT (SCHED_FIFO o SCHED_RR)
  - No esgotar recursos CPU! (while true)
- Bloquejar memòria de paginació (mlockall)
  - Cal “tocar” pàgines, si no Linux no les reserva!
- Procés RT utilitza *buffers* intermedis
  - Procés no RT gestiona escriptura / lectura
- PI Mutex (PTHREAD_PRIO_INHERIT)
- Temporitzadors d'alta resolució (HRT)

Influència del maquinari

- Competició CPU vs DMA
  - Drivers DMA-less, menys rendiment
- Estalvi d'energia (CSTATE)
  - processor.max_state=1 i idle=poll
  - /dev/cpu_dma_latency
- System Management Interrupts (SMI)
  - Mesurar amb *hwlatdetect*
- NO utilitzar consola VGA (>100ms!)

Justificació *benchmarking*

- **Modelatge Linux altament complex**
  - 15 MLOC (Millions Lines of Code)
  - *Release* cada 2 mesos
- **Modelatge maquinari modern també**
  - Pipeline, hyperthreading, cache...
- **Avantatges *benchmarking***
  - Fàcil d'implementar / reproduir
  - A l'abast de tothom

Metodologia

- Passos
  1. Definir propietats experiment segons objectius
  2. Selecció de mètriques i descriptors estadístics
  3. Avaluació, selecció i adaptació d'eines
  4. Validació i ajust de càrregues
  5. Disseny i implementació de l'experiment
  6. Execució sobre 4 configuracions de Linux
  7. Tractament, anàlisi de resultats

Propietats experiment

- **Nivell d'estrès notable**
  - Mesurar afectació CPU, memòria, disc..
- **Freqüència dels esdeveniments**
  - F > F_Pijtor_Cas_Desitjat x1/2 (*rule of thumb*)
  - 1/500us ~ 1/200us x 1/2 (200us mesures OSADL)
- **Duració de l'experiment**
  - Recomanat OSADL 10^9 (serien 5,8 dies a 500us!)
  - Execució de 8h per falta de temps

Mètriques i estadístics

- **Latència:**
  - Percentils (99% *soft RT*, 100% *hard RT*)
  - Desviació típica (determinisme)
- **Rendiment:**
  - Penalized = T_Exec / T_Millor_Exec
  - Slowdown = (T_Exec + T_Espera) / T_Exec
    - Scheduler i No-Scheduler (maj. I/O)
  - Pearson = Dependència variables mesurades

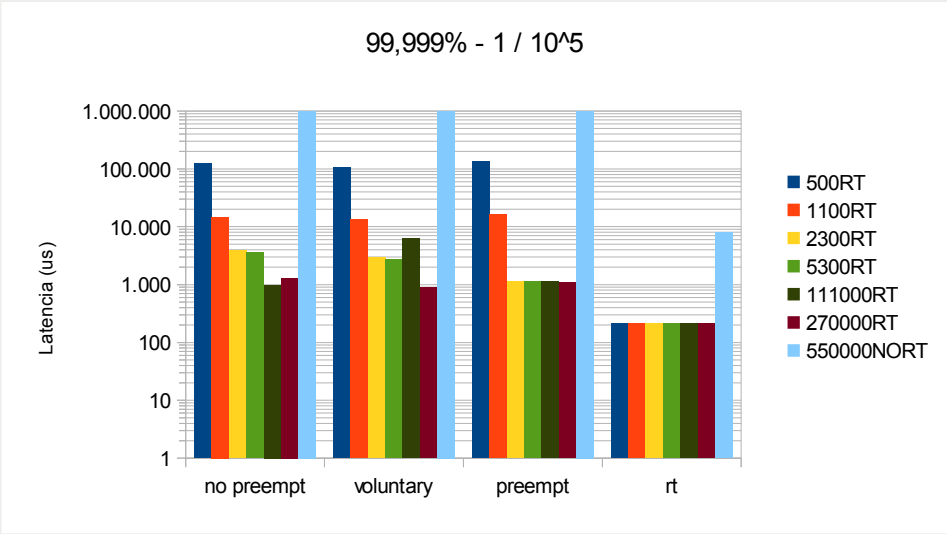
Eines de mesura i càrrega

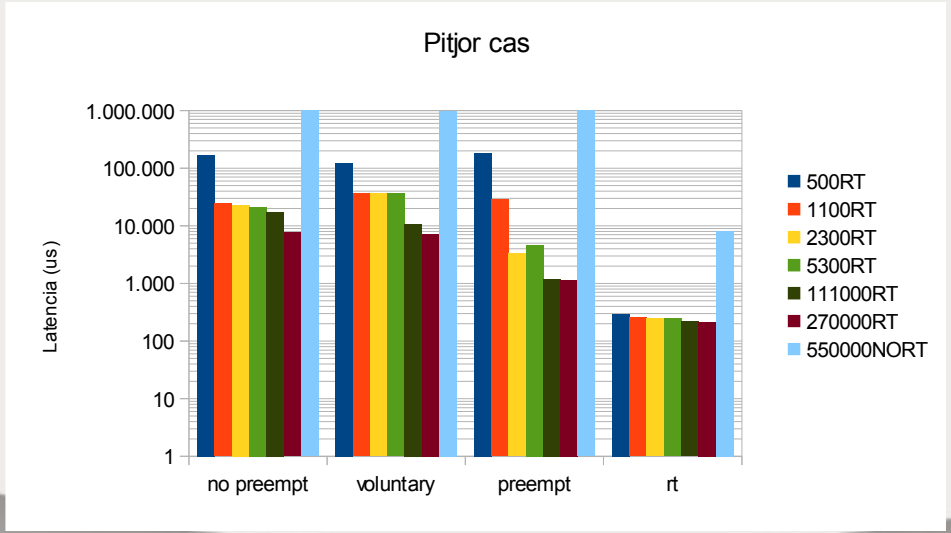
- **Cyclictest (latència):**
  1. Registrar estampa de temps DORMIR
  2. Programar un despertador i dormir
  3. Latència = DESPERTAR - DORMIR
- **GNU time modificat (rendiment)**
  - Temps: execució i espera
  - Canvis de context: voluntaris i involuntaris
  - Page fault: major i minor

Càrregues de l'experiment

- Netperf, ab (*benchmark* de xarxa)
  - Estrès CPU + codi nucli
- Iozone (*benchmark* de disc)
  - Estrès camins E/S
- Stress (estrès varis)
  - Estrès memòria
- Hackbench (*benchmark* planificador)
  - Estrès planificador

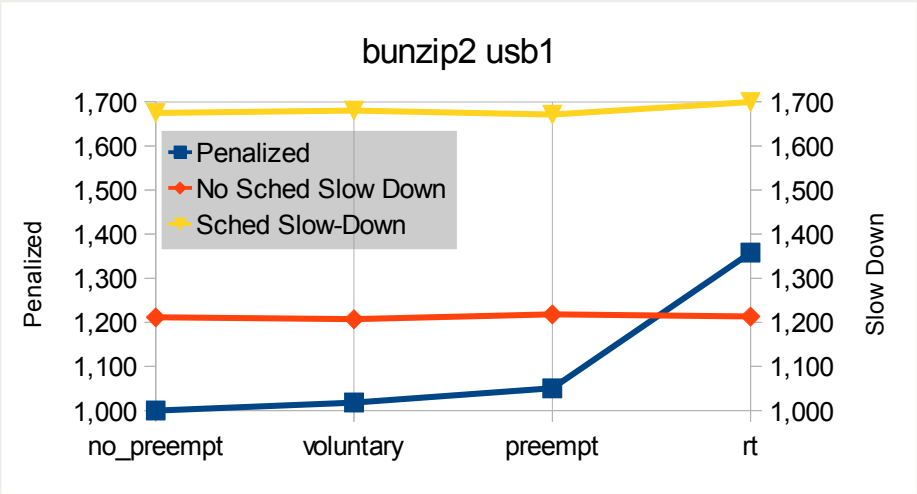
Assoliment del temps real

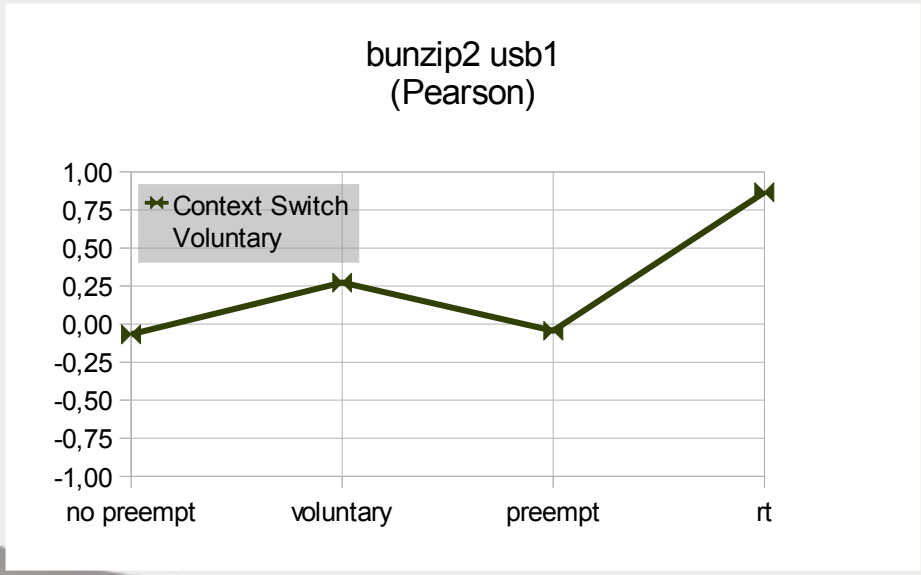




- Llegenda xxxRT
  - F(Desperta) = xxx us
  - RT/NORT = temps real
- **No Patch RT > 1100RT**
  - 99,999% ~ 1-10ms
  - Worst ~ 10-100ms
- **Patch RT**
  - Worst ~ 200us!

Impacte en el rendiment





- Penalització (*penalized*)
  - Lineal fins *preempt*
  - **RT: +35%!**
- Ralentització(*slowdown*)
  - *Constant fins preempt*
  - RT: +Scheduler
- Pearson (CS-V)
  - **Preempció sec. crítiques!**

Conclusions

- Linux 2.6/3 + Intel ATOM @ 1,60Ghz (UP)
  - PREEMPT (F ≤ 1/2300us)
    - Soft RT 99,999% < 1-10ms
    - Penalització rendiment 5%
  - PATCH RT (F ≤ 1/500us)
    - Hard RT Worst < 220us
    - Penalització rendiment 35%!
    - Causa: preempció seccions crítiques

Línies obertes

- Verificar resultats amb altres càrregues i equips.
- Esbrinar orígens latència PREEMPT amb FTRACE.
- Avaluar PREEMPT amb *threadirqs*.
- Avaluar 3.6.4-rt11 (Octubre 2012) *lazy preemption* millora cost preempció?
- Avaluar multiprocessador (SMP), afinitats...
- Realitzar estudi influència maquinari diferents arquitectures (x86, PowerPC, ARM, MIPS...)