Memoria compartida distribuida

Ricardo Jiménez Peris, Marta Patiño Martínez y Ernesto Jiménez

Distributed
Systems
Laboratory
Universidad Politécnica de Madrid (UPM)
http://lsd.ls.fi.upm.es/lsd/lsd.htm

La producción de este material ha sido financiada parcialmente por
Microsoft Research Cambridge (Award MS-2004-393)

Índice

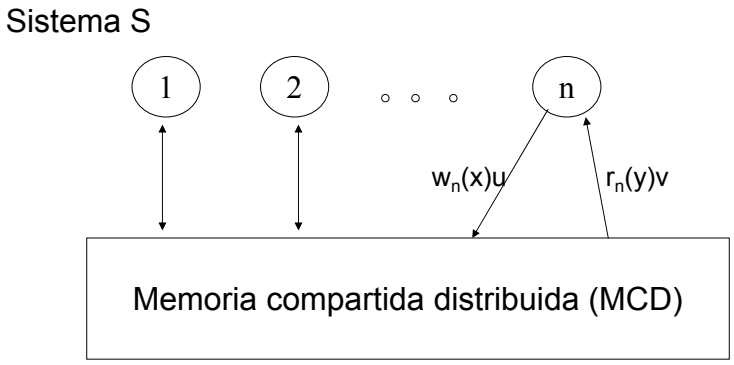
Bibliografía

- M. Ahamad, G. Neiger, J.E. Burns, P. Kohli y P.W. Hutto. Causal Memory: Definitions, implementation and programming. Distributed Computing, 9(1). 37-49. 1995
- E. Jiménez, A. Fernández y V. Cholvi. A Parametrized Algorithm that Implements Sequential, Causal, and Cache Memory Consistency. Euro PDP. 437-444. 2002.
- L. Lamport. How to make multiprocessor computer that correctly executes multiprocess programs. IEEE Transactions on Computers, 28(9). 690-691. 1979.
- L. Lamport. On interprocess communication: Parts I and II. Distributed Computing, 1(2). 77-101. 1986.
- K. Li and P. Hudak. Memory Coherence in Shared Virtual Memory Systems. ACM Transactions on Computer Systems. 7(4). 321-359. 1989.
- R.J. Lipton y J.S. Sandberg. PRAM: A scalable shared memory. CS-TR-180-88, Princeton University. 1988.

Definición de MCD.

- Un memoria compartida distribuida (MCD) es una abstracción de una memoria utilizada, mediante operaciones de lectura y escritura, por todos los procesos de un sistema distribuido.
- Sistema de MCD: conjunto formado por todos los procesos y la MCD.
- El sistema distribuido es *asíncrono* y *sin fallos*. Tenemos un conjunto de procesos todos correctos e interconectados mediante canales bidireccionales fiables.
- Modelo de coherencia: semántica de las operaciones de lectura y escritura sobre la MCD.

Definición de MCD (cont.)



Modelo coherencia: orden en la ejecución de las operaciones de lectura y escritura sobre la MCD

Modelo atómico.

Coherencia atómica o linealizable [Lamport 1986]:

- El resultado de las operaciones de memoria en cualquier ejecución es el mismo que el que se produciría sí:
  1. Todas las operaciones fueran ejecutadas en el orden real en el que se produjeron.
  2. Las operaciones de cada proceso siguen el mismo orden que el especificado dentro del programa.

Modelo atómico (cont.)

Para una ejecución R, sea α el conjunto de operaciones producidas en R.

- *Orden en un proceso* ($op <_p op'$): Si op es invocada por el proceso p antes que op´.
- *Orden en tiempo-real* ($op <_{rt} op'$): Si op termina de ejecutarse antes de que op´ comience a ser ejecutada.

β_{atom} (*vista atómica*) : una ordenación de todas las operaciones de α que preserve $<_p$ y $<_{rt}$

Modelo atómico (cont.)

β_{atom} es *legal*:

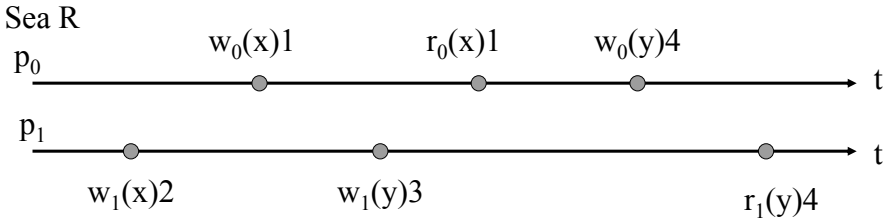
Si para toda $op=r_p(x)u$:

1. Existe en β_{atom} una $op'=w_q(x)u$ tal que op´precede a op.
2. No existe en β_{atom} una $op''=w_s(x)v$ tal que op´ precede a op´´ que precede a op.

Un sistema es atómico (tiene coherencia atómica): si toda ejecución R de S existe una β_{atom} legal.

Modelo atómico (cont.)

Sea R



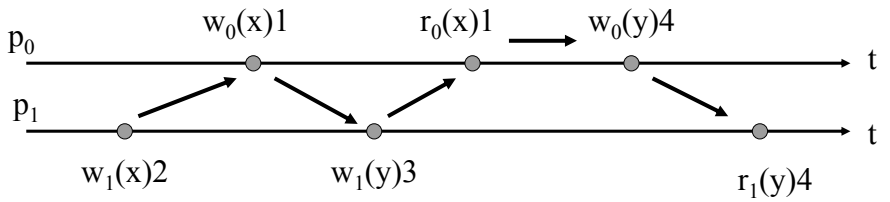
Vista atómica (β_{atom}) :

$w_1(x)2$ $w_0(x)1$ $w_1(y)3$ $r_0(x)1$ $w_0(y)4$ $r_1(y)4$

R es una ejecución atómica: existe una vista atómica legal

Modelo atómico (cont.)

Dependencias atómicas



Vista atómica (β_{atom}) :

$w_1(x)2$ $w_0(x)1$ $w_1(y)3$ $r_0(x)1$ $w_0(y)4$ $r_1(y)4$

Modelo atómico (cont.)

- Ventaja: Sigue el paradigma de programación tradicional.
- Desventaja: Latencia alta en las operaciones cuando existe un número elevado de procesos en el sistema.

¿Se pueden relajar la semántica? → *Modelo secuencial*

Algoritmo atómico

[Li & Hudak 1989]

function $w_p(x)u$:
send (WRITE,x,u) to serv
wait until receive (ACK-W) from serv

function $r_p(x)$:
send (READ,x) to serv
wait until receive (ACK-R,$value_{serv}$) from serv
return ($value_{serv}$)

(Código proceso p)

$M_{serv}[x] \leftarrow \perp, \forall x$ // inicialmente
repeat forever
wait until receive (op_q,var_q,$value_q$) from q
if op_q=WRITE **then**
$M_{serv}[var_q] \leftarrow value_q$
send (ACK-W) to q
else
$value_{serv} \leftarrow M_{serv}[var_q]$
send (ACK-R, $value_{serv}$) to q
endrepeat

(Código proceso servidor *serv*)

Algoritmo atómico (cont.)

Corrección del algoritmo para coherencia atómica:

Sea β_{atom} la ordenación de todas las operaciones de α en cada R según se ejecutan en el proceso serv.

- Esta β_{atom} preserva el orden de las operaciones en cada proceso y el orden en tiempo real (i.e., $<_p$ y $<_{rt}$).
- Esta β_{atom} es legal:
  - Si existe op=r(x)u es porque previamente se hizo op´=w(x)u (inicialmente se supone que w(x)=⊥)
  - No existe op´´=w(x)v entre medias de op´ y op, porque en ese caso serv leería v y no u.

Modelo secuencial.

Coherencia secuencial [Lamport 1979]:

- El resultado de las operaciones de memoria en cualquier ejecución es el mismo que el que se produciría sí:
  1. Todas las operaciones fueran ejecutadas en un orden secuencial (no tiene por qué ser el real).
  2. Las operaciones de cada proceso siguen el mismo orden que el especificado dentro del programa.

Modelo secuencial (cont.)

Para una ejecución R, sea α el conjunto de operaciones producidas en R.

- *Orden en un proceso* ($op <_p op'$): Si op es invocada por el proceso p antes que op´.
- *Orden de ejecución* ($op < op'$): Si ocurre alguno de los siguientes casos:
  1. op y op´ son invocadas por p, $op <_p op'$.
  2. op=w(x)v y op´=r(x)v.
  3. Si existe op´´ tal que $op < op'' < op$

Modelo secuencial (cont.)

β_{sec} (*vista secuencial*) : una ordenación de todas las operaciones de α que preserve $<_p$ y $<$

β_{sec} es *legal*:

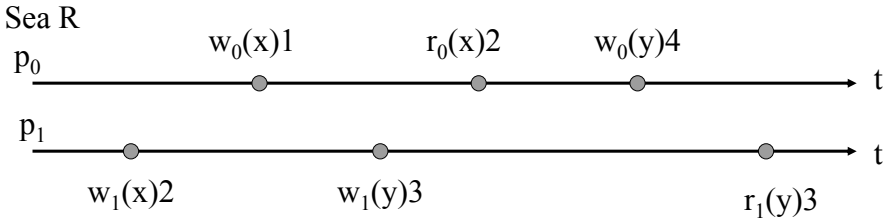
Si para toda $op=r_p(x)u$:

1. Existe en β_{sec} una $op'=w_q(x)u$ tal que op´ precede a op.
2. No existe en β_{sec} una $op''=w_s(x)v$ tal que op´precede a op´´ que precede a op.

Un sistema es secuencial (tiene coherencia secuencia): si para toda ejecución R de S existe una β_{sec} legal.

Modelo secuencial (cont.)

Sea R



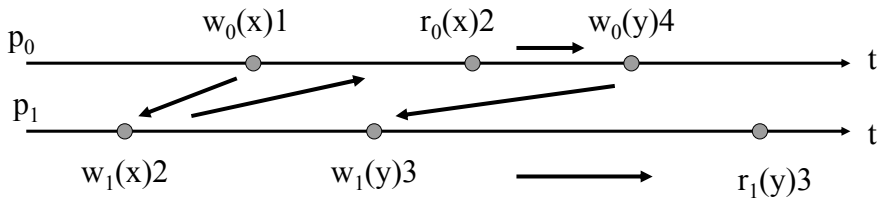
Vista atómica (β_{sec}) :

$w_0(x)1$ $w_1(x)2$ $r_0(x)2$ $w_0(y)4$ $w_1(y)3$ $r_1(y)3$

R es una ejecución secuencial: existe una vista secuencial legal

Modelo secuencial (cont.)

Dependencias secuenciales



Vista atómica (β_{sec}) :

$w_0(x)1$ $w_1(x)2$ $r_0(x)2$ $w_0(y)4$ $w_1(y)3$ $r_1(y)3$

Modelo secuencial (cont.)

Técnicas para reducir latencia en las operaciones.

- Replicación de datos.
- Actualización de las copias:
  1. Invalidación de las copias obsoletas.
  2. Propagación de los nuevos valores.

Modelo secuencial (cont.)

- Ventaja: Al igual que el modelo atómico, sigue el paradigma de programación tradicional.
- Desventaja: Latencia alta en las operaciones cuando existe un número elevado de procesos en el sistema. Es imposible que todas las operaciones sean "rápidas" (i.e., que en todos los casos devuelvan la copia local de la variable sin "sincronizarse" con otros procesos)

¿Se pueden relajar aún más la semántica? → *Modelo causal*

Algoritmo secuencial

[Jiménez et al 2002]

$M_p[x] \leftarrow \perp, \forall x$ // inicialmente

function $w_p(x)u$:
  $M_p[x] \leftarrow u$
  include (x,u) **to** $updates_p$

function $r_p(x)$:
  if ($updates_p \neq \emptyset$)
    and ((x,-) $\notin$ $updates_p$) **then**
      wait until $turn_p$=p
  return ($M_p[x]$)

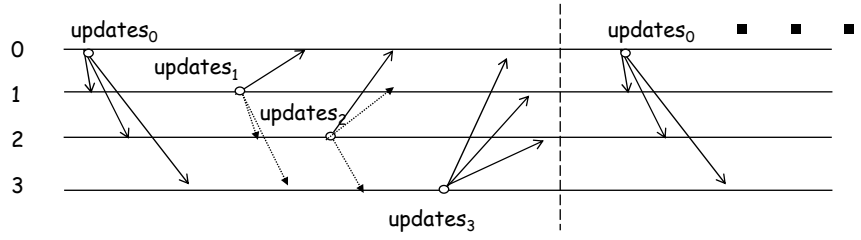
(Código proceso p)

when $turn_p$=p: //p propaga updates
  send ($updates_p$) to all
  $updates_p \leftarrow \emptyset$
  $turn_p \leftarrow (turn_p + 1) \bmod n$

when $turn_p \neq p$ y recibido ($updates_q$):
  //p aplica updates
  while updates $\neq \emptyset$ **do**
    extract (x,u) **from** $updates_q$
    if (x,-) $\notin$ $updates_p$) **then**
      $M_p[x] \leftarrow u$
  $turn_p \leftarrow (turn_p + 1) \bmod n$

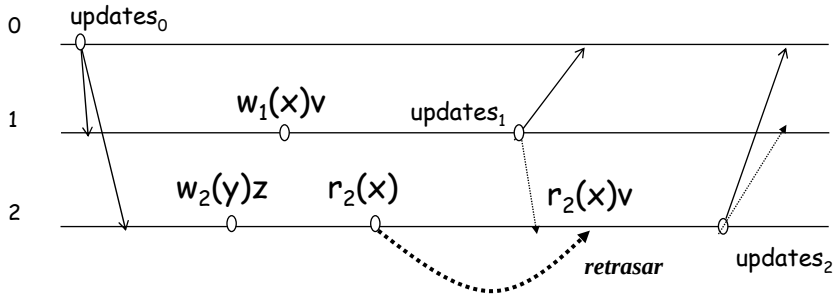
Algoritmo secuencial (cont.)

- Impone un mismo orden en el envío de escrituras



Algoritmo secuencial (cont.)

- Se impone en p un mismo orden en la aplicación de las escrituras de q
- Se retrasa en p una $r_p(x)$ si no es su turno y p previamente no $w_p(x)$



Algoritmo secuencial (cont.)

Corrección del algoritmo para coherencia secuencial:

Sea β_{sec} la ordenación de todas las operaciones de α en cada R según se aplican las operaciones de escritura en los procesos, y las operaciones de lectura justo detrás de la escritura cuyo valor leen.

- Esta β_{sec} preserva el orden de las operaciones en cada proceso y el orden secuencial (i.e., $<_p$ y $<$).
- Esta β_{sec} es legal:
  - Si existe op=r(x)u es porque previamente se hizo op´=w(x)u (inicialmente se supone que w(x)=⊥)
  - No existe op´´=w(x)v entre medias de op´ y op, porque en ese caso serv leería v y no u.

Modelo causal.

Coherencia causal [Ahamad et al 1995]:

- En cada ejecución R, *sólo* las operaciones relacionadas "causalmente" (i.e., aquellas por cuya causa haya surtido efecto en otras) tienen que ser percibidas por todos los procesos siguiendo ese mismo orden causal.

Ejemplo: Si existe $r_p(x)u$ en R, es porque también debe existir $w_q(x)u$. Por tanto, esta escritura debe preceder en el orden causal a esta lectura.

Modelo causal (cont.)

Para una ejecución R, sea α_p el subconjunto de operaciones de α producidas en R tal que eliminamos de α todas las operaciones de lectura generadas por un proceso distinto de p.

- *Orden en un proceso* ($op <_p op'$): Si op es invocada por el proceso p antes que op´.
- *Orden de ejecución* ($op < op'$): Si ocurre alguno de los siguientes casos:
  1. op y op´ son invocadas por p, $op <_p op'$.
  2. op=w(x)v y op´=r(x)v.
  3. Si existe op´´ tal que $op < op'' < op$

Modelo causal (cont.)

$\beta_{cau\text{-}p}$ (*vista causal para p*) : una ordenación de todas las operaciones de α_p que preserve $<_p$ y $<$

$\beta_{cau\text{-}p}$ es *legal*:

Si para toda $op=r_p(x)u$ perteneciente a α_p:

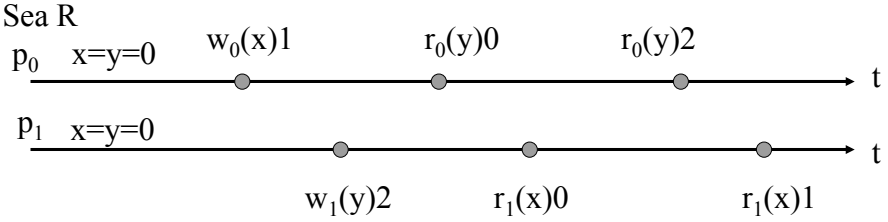
1. Existe en $\beta_{cau\text{-}p}$ una $op' = w_q(x)u$ tal que op´ precede a op.
2. No existe en $\beta_{cau\text{-}p}$ una $op'' = w_s(x)v$ tal que op´precede a op´´ que precede a op.

Distintos procesos pueden tener distintas vistas causales legales, ya que α_p y α_q son diferentes.

Un sistema es causal (tiene coherencia causal): si para toda ejecución R de S existe una $\beta_{cau\text{-}p}$ legal para todo proceso p.

Modelo causal (cont.)

Sea R



Vistas causales

$\beta_{cau\text{-}0}$: $w_0(x)1$ $r_0(y)0$ $w_1(y)2$ $r_0(y)2$

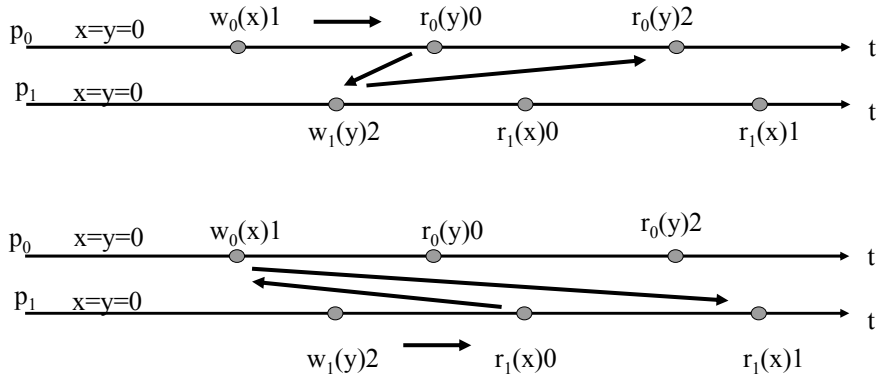
$\beta_{cau\text{-}1}$: $w_1(y)2$ $r_1(x)0$ $w_0(x)1$ $r_1(x)1$

R es una ejecución causal: existe una vista causal legal para 0 y 1

Modelo causal (cont.)

Dependencias causales

$w_0(x)1$	$\rightarrow$ $w_1(y)2$	*para* p_0
$w_1(y)2$	$\rightarrow$ $w_0(x)1$	*para* p_1



Modelo causal (cont.)

- Ventaja:
  - Latencia baja incluso cuando existe un número elevado de procesos en el sistema. Todas las operaciones pueden ser "rápidas".
  - Algunos programas se pueden ejecutar directamente en sistemas causales dando los mismos resultados que con coherencia secuencial.
- Desventaja: Hay que cambiar en la mayoría de los casos el paradigma de programación tradicional.

¿Se pueden relajar aún más la semántica? → *Modelo pRAM*

Algoritmo causal

[Jiménez et al 2002]

$M_p[x] \leftarrow \perp, \forall x$ // inicialmente

function $w_p(x)u$:
  $M_p[x] \leftarrow u$
  include (x,u) **to** $updates_p$

function $r_p(x)$:
  return ($M_p[x]$)

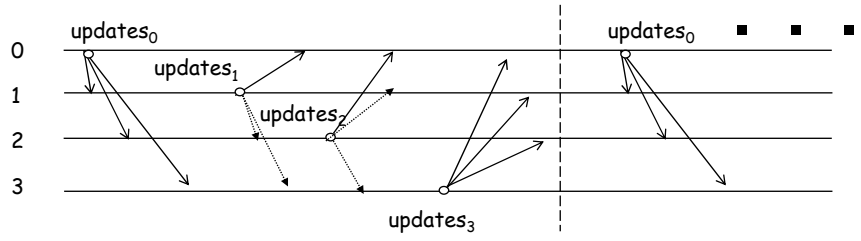
(Código proceso p)

when $turn_p$=p: //p propaga updates
  send ($updates_p$) to all
  $updates_p \leftarrow \emptyset$
  $turn_p \leftarrow (turn_p + 1) \bmod n$

when $turn_p \neq p$ y recibido ($updates_q$):
  //p aplica updates
  while updates $\neq \emptyset$ **do**
    extract (x,u) **from** $updates_q$
    $M_p[x] \leftarrow u$
  $turn_p \leftarrow (turn_p + 1) \bmod n$

Algoritmo causal (cont.)

- Impone un mismo orden en el envío de escrituras



Algoritmo causal (cont.)

Corrección del algoritmo para coherencia causal:

Sea $\beta_{cau\text{-}p}$ la ordenación de todas las operaciones de α_p en cada R según se generan las lecturas y escrituras de p, y según se aplican las operaciones de escritura de los demás procesos en p.

- Esta β_{sec} preserva el orden de las operaciones en cada proceso y el orden secuencial (i.e., $<_p$ y $<$).
- Esta β_{sec} es legal:
  - Si existe op=r(x)u es porque previamente se hizo op´=w(x)u (inicialmente se supone que w(x)=⊥)
  - No existe op´´=w(x)v entre medias de op´ y op, porque en ese caso serv leería v y no u.

Modelo pRAM.

Coherencia pRAM (*pipelined RAM*) [Lipton & Sandberg 1988]:

- En cada ejecución R, <u>*sólo*</u> las operaciones de escritura de todo proceso *q* deben de ser percibidas en cada proceso *p* en el mismo orden en el que fueron invocadas por *q*.

Modelo pRAM (cont.)

Para una ejecución R, sea α_p el subconjunto de operaciones de α producidas en R tal que eliminamos de α todas las operaciones de lectura generadas por un proceso distinto de p.

- *Orden en un proceso* ($op <_p op'$): Si op es invocada por el proceso p antes que op´.

Modelo pRAM (cont.)

$\beta_{pRAM\text{-}p}$ (*vista pRAM para p*) : una ordenación de todas las operaciones de α_p que preserve $<_p$

$\beta_{pRAM\text{-}p}$ es *legal*:

Si para toda $op=r_p(x)u$ perteneciente a α_p:

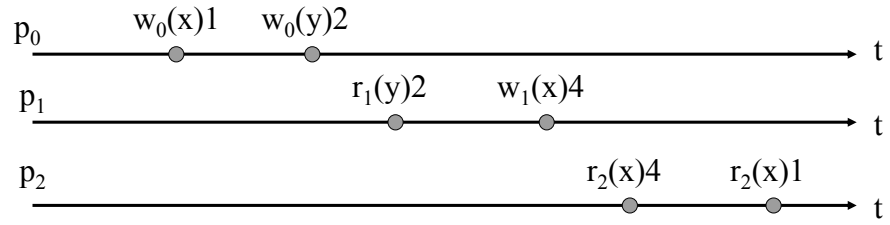
1. Existe en $\beta_{pRAM\text{-}p}$ una $op'=w_q(x)u$ tal que op´ precede a op.
2. No existe en $\beta_{pRAM\text{-}p}$ una $op''=w_s(x)v$ tal que op´precede a op´´ que precede a op.

Distintos procesos pueden tener distintas vistas pRAM legales, ya que α_p y α_q son diferentes.

Un sistema es pRAM (tiene coherencia pRAM): si para toda ejecución R de S existe una $\beta_{pRAM\text{-}p}$ legal para todo proceso p.

Modelo pRAM (cont.)

Sea R



Vistas pRAM

(una de las posibles) $\beta_{pRAM\text{-}0}$: $w_0(x)1$ $w_0(y)2$ $w_1(x)4$

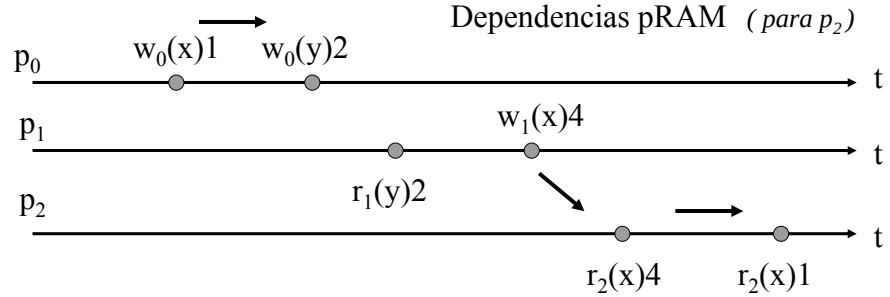
$\beta_{pRAM\text{-}1}$: $w_0(x)1$ $w_0(y)2$ $r_1(y)2$ $w_1(x)4$

(una de las posibles) $\beta_{pRAM\text{-}2}$: $w_1(x)4$ $r_2(x)4$ $w_0(x)1$ $w_0(y)2$ $r_2(x)1$

R es una ejecución pRAM: existe una vista pRAM legal para 0, 1 y 2

Modelo pRAM (cont.)

Dependencias pRAM *(para p_2)*



Vista pRAM de 2

$\beta_{pRAM\text{-}2}$: $w_1(x)4$ $r_2(x)4$ $w_0(x)1$ $w_0(y)2$ $r_2(x)1$

Hay vista pRAM legal pero no causal para 2

Modelo pRAM (cont.)

- Ventaja: Latencia baja incluso cuando existe un número elevado de procesos en el sistema. Todas las operaciones pueden ser "rápidas".
- Desventaja:
  - Hay que cambiar el paradigma de programación tradicional.
  - No todos los programas que requieren coherencia secuencial se podrían ejecutar (ni transformándolos) en sistemas pRAM.

Relación entre modelos de memoria