No interferencia y disponibilidad en una plataforma certificada de virtualización

Gustavo Betarte

Grupo de Seguridad Informática
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
www.fing.edu.uy/~gustun

JIAP 2011

Plan

Grupo de Seguridad Informática (GSI - FING)

- Formado a comienzos del año 2006
- Integrado por docentes y profesionales del InCo, IIE y la URI de la Facultad de Ingeniería
- Objetivos
  - Formación de RRHH (grado y posgrado)
  - Investigación
  - Asesoramiento especializado

Cursos de grado y posgrado

- **Fundamentos de la Seguridad Informática**
  - Curso electivo de grado (Ing. en Computación)
  - Curso de posgrado (Pedeciba Informática)

Cursos de grado y posgrado

- **Fundamentos de la Seguridad Informática**
  - Curso electivo de grado (Ing. en Computación)
  - Curso de posgrado (Pedeciba Informática)
- **Seguridad de Sistemas**, **Seguridad de Aplicaciones**, **Gestión de la Seguridad de la Información**
  - Cursos del diploma de especialización del Centro de Posgrados y Actualización Profesional (CPAP)

Cursos de grado y posgrado

- **Fundamentos de la Seguridad Informática**
  - Curso electivo de grado (Ing. en Computación)
  - Curso de posgrado (Pedeciba Informática)
- **Seguridad de Sistemas**, **Seguridad de Aplicaciones**, **Gestión de la Seguridad de la Información**
  - Cursos del diploma de especialización del Centro de Posgrados y Actualización Profesional (CPAP)
- **Taller de Seguridad Informática**
  - Curso electivo de grado (Ing. en Computación)
  - Focalizado en aplicación de metodologías y uso de herramientas

Áreas

- Especificación y verificación formal de arquitecturas de seguridad de sistemas críticos

Áreas

- Especificación y verificación formal de arquitecturas de seguridad de sistemas críticos
- Ambientes para el entrenamiento en Seguridad Informática

Áreas

- Especificación y verificación formal de arquitecturas de seguridad de sistemas críticos
- Ambientes para el entrenamiento en Seguridad Informática
- Análisis estático y dinámico de seguridad de aplicaciones web

Áreas

- Especificación y verificación formal de arquitecturas de seguridad de sistemas críticos
- Ambientes para el entrenamiento en Seguridad Informática
- Análisis estático y dinámico de seguridad de aplicaciones web
- Gestión de Seguridad de la Información

Áreas

- Especificación y verificación formal de arquitecturas de seguridad de sistemas críticos
- Ambientes para el entrenamiento en Seguridad Informática
- Análisis estático y dinámico de seguridad de aplicaciones web
- Gestión de Seguridad de la Información
- Análisis Forense Digital

Áreas

- Especificación y verificación formal de arquitecturas de seguridad de sistemas críticos
- Ambientes para el entrenamiento en Seguridad Informática
- Análisis estático y dinámico de seguridad de aplicaciones web
- Gestión de Seguridad de la Información
- Análisis Forense Digital
- Gestión de Identidad Electrónica

Proyecto VirtualCert
Hacia una Plataforma de Virtualización Certificada

- Modelo idealizado de virtualización

Proyecto VirtualCert

Hacia una Plataforma de Virtualización Certificada

- Modelo idealizado de virtualización
- Prueba formal de propiedades de no interferencia entre sistemas guests de la plataforma

Proyecto VirtualCert

Hacia una Plataforma de Virtualización Certificada

- Modelo idealizado de virtualización
- Prueba formal de propiedades de no interferencia entre sistemas guests de la plataforma
- Especificación formal y verificación de corrección usando el asistente de pruebas Coq

Proyecto VirtualCert

Hacia una Plataforma de Virtualización Certificada

- Modelo idealizado de virtualización
- Prueba formal de propiedades de no interferencia entre sistemas guests de la plataforma
- Especificación formal y verificación de corrección usando el asistente de pruebas Coq
- Derivación de una especificación ejecutable e implementación en C de un hipervisor que garantice las propiedades de seguridad

Proyecto VirtualCert

Hacia una Plataforma de Virtualización Certificada

- Modelo idealizado de virtualización
- Prueba formal de propiedades de no interferencia entre sistemas guests de la plataforma
- Especificación formal y verificación de corrección usando el asistente de pruebas Coq
- Derivación de una especificación ejecutable e implementación en C de un hipervisor que garantice las propiedades de seguridad
- Proyecto ANII, Fondo Clemente Estable Ed. 2009:
  - Gilles Barthe, IMDEA Software (España)
  - Gustavo Betarte, Juan Diego Campo, Carlos Luna (GSI - FING)

Background: Verificación de Sistemas Operativos

- Verificación de SO desde 1970
- Grandes avances en tecnología de pruebas formales
- Verificación de lenguajes de programación ya es una realidad: verificación de SO es la próxima frontera

Background: Verificación de Sistemas Operativos

- Verificación de SO desde 1970
- Grandes avances en tecnología de pruebas formales
- Verificación de lenguajes de programación ya es una realidad: verificación de SO es la próxima frontera
- Proyectos insignia:

Background: Verificación de Sistemas Operativos

- Verificación de SO desde 1970
- Grandes avances en tecnología de pruebas formales
- Verificación de lenguajes de programación ya es una realidad: verificación de SO es la próxima frontera
- Proyectos insignia:
  - L4.verified: verificación formal del exokernel seL4

Background: Verificación de Sistemas Operativos

- Verificación de SO desde 1970
- Grandes avances en tecnología de pruebas formales
- Verificación de lenguajes de programación ya es una realidad: verificación de SO es la próxima frontera
- Proyectos insignia:
  - L4.verified: verificación formal del exokernel seL4
  - Hyper-V: verificación formal del hipervisor de Microsoft

Background: Verificación de Sistemas Operativos

- Verificación de SO desde 1970
- Grandes avances en tecnología de pruebas formales
- Verificación de lenguajes de programación ya es una realidad: verificación de SO es la próxima frontera
- Proyectos insignia:
  - L4.verified: verificación formal del exokernel seL4
  - Hyper-V: verificación formal del hipervisor de Microsoft
- Lógicas de programación para razonar sobre código de bajo nivel:
  - FLINT
  - Separation Logic
  - Verve

Motivación y desafíos

- Foco principal de L4.verified e Hyper-V es en corrección formal

Motivación y desafíos

- Foco principal de L4.verified e Hyper-V es en corrección formal
- Propiedades no funcionales son igualmente importantes

Motivación y desafíos

- Foco principal de L4.verified e Hyper-V es en corrección formal
- Propiedades no funcionales son igualmente importantes
  - Confidencialidad e Integridad
    - Plataformas de virtualización deben garantizar aislamiento
    - Evaluaciones de seguridad (CC)

Motivación y desafíos

- Foco principal de L4.verified e Hyper-V es en corrección formal
- Propiedades no funcionales son igualmente importantes
  - Confidencialidad e Integridad
    - Plataformas de virtualización deben garantizar aislamiento
    - Evaluaciones de seguridad (CC)
  - Disponibilidad
    - Plataformas de virtualización deben respetar restricciones de disponibilidad
    - Organismos de Certificación (DO178)

Motivación y desafíos

- Foco principal de L4.verified e Hyper-V es en corrección formal
- Propiedades no funcionales son igualmente importantes
  - Confidencialidad e Integridad
    - Plataformas de virtualización deben garantizar aislamiento
    - Evaluaciones de seguridad (CC)
  - Disponibilidad
    - Plataformas de virtualización deben respetar restricciones de disponibilidad
    - Organismos de Certificación (DO178)
- Mas allás de propiedades de *safety*
  - Propiedades de aislamiento son propiedades 2-safety
  - Propiedades de disponibilidad son propiedades de *liveness*

Motivación y desafíos

- Foco principal de L4.verified e Hyper-V es en corrección formal
- Propiedades no funcionales son igualmente importantes
  - Confidencialidad e Integridad
    - Plataformas de virtualización deben garantizar aislamiento
    - Evaluaciones de seguridad (CC)
  - Disponibilidad
    - Plataformas de virtualización deben respetar restricciones de disponibilidad
    - Organismos de Certificación (DO178)
- Mas allás de propiedades de *safety*
  - Propiedades de aislamiento son propiedades 2-safety
  - Propiedades de disponibilidad son propiedades de *liveness*

VirtualLogix (ahora Red Bend Software)

- Proporcionó requerimientos informales en etapas iniciales del trabajo
- Foco sugerido: plataformas de paravirtualización *Xen-like*

Hipervisores

- Permiten que distintos SO coexistan sobre una misma plataforma de hardware

Hipervisores

- Permiten que distintos SO coexistan sobre una misma plataforma de hardware
- Proveen soporte para que distintas aplicaciones ejecuten sobre los SO *guests*

Hipervisores

- Permiten que distintos SO coexistan sobre una misma plataforma de hardware
- Proveen soporte para que distintas aplicaciones ejecuten sobre los SO *guests*
- Permiten que aplicaciones con diferentes políticas de seguridad pueden ejecutar concurrentemente en forma segura

Hipervisores

- Permiten que distintos SO coexistan sobre una misma plataforma de hardware
- Proveen soporte para que distintas aplicaciones ejecuten sobre los SO *guests*
- Permiten que aplicaciones con diferentes políticas de seguridad pueden ejecutar concurrentemente en forma segura
- Son actualmente fuertemente usados como mecanismos para mejorar la seguridad y flexibilidad de plataformas computacionales

Hipervisores

- Permiten que distintos SO coexistan sobre una misma plataforma de hardware
- Proveen soporte para que distintas aplicaciones ejecuten sobre los SO *guests*
- Permiten que aplicaciones con diferentes políticas de seguridad pueden ejecutar concurrentemente en forma segura
- Son actualmente fuertemente usados como mecanismos para mejorar la seguridad y flexibilidad de plataformas computacionales
- Se prevé que serán la base computacional de CPDs y *cloud computing*

Hipervisores

- Permiten que distintos SO coexistan sobre una misma plataforma de hardware
- Proveen soporte para que distintas aplicaciones ejecuten sobre los SO *guests*
- Permiten que aplicaciones con diferentes políticas de seguridad pueden ejecutar concurrentemente en forma segura
- Son actualmente fuertemente usados como mecanismos para mejorar la seguridad y flexibilidad de plataformas computacionales
- Se prevé que serán la base computacional de CPDs y *cloud computing*

Hipervisores son un objetivo prioritario de especificación y verificación formal

Modelos Idealizados vs. implementaciones

Razonar sobre implementaciones

- Provee garantías más sólidas
- Es factible para *algunos* exokernels e hipervisores
- Puede ser factible para *algunas* propiedades básicas de *algunos* sistemas
- En general es impracticable (Linux Kernel)
- Puede no ser requerido por procesos de evaluación

Modelos Idealizados vs. implementaciones

Razonar sobre implementaciones

- Provee garantías más sólidas
- Es factible para *algunos* exokernels e hipervisores
- Puede ser factible para *algunas* propiedades básicas de *algunos* sistemas
- En general es impracticable (Linux Kernel)
- Puede no ser requerido por procesos de evaluación

Modelos idealizados

- Muchos detalles concernientes al comportamiento de un SO son irrelevantes para la seguridad
- Modelos idealizados pueden proveer un nivel adecuado de abstracción. Las pruebas son más focalizadas y pueden ser desarrolladas en tiempo razonable
- Desventajas: modelos idealizados pueden no capturar todos los detalles relevantes.

Plataforma Virtualizada

Visión abstracta

- Particionado de memoria
- No es fija: allocation & deallocation
- No es total: puede haber memoria que no pertenezca a ningún SO

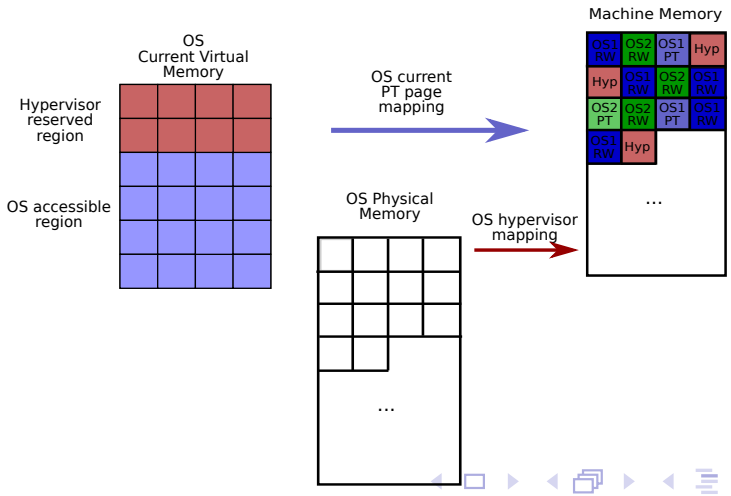
Plataforma Virtualizada

Visión abstracta

- Particionado de memoria
- No es fija: allocation & deallocation
- No es total: puede haber memoria que no pertenezca a ningún SO

Modelo idealizado

- Direcciones: va, pa y ma
- Mappings entre direcciones
- Páginas contienen valores RW o tablas de páginas



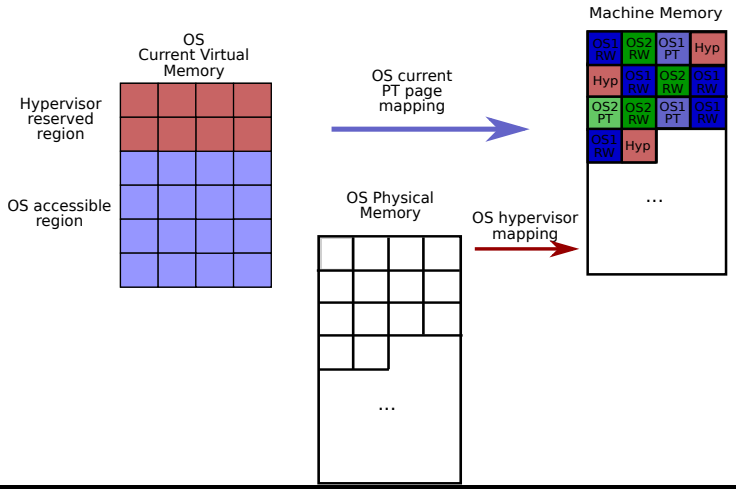
Plataforma Virtualizada

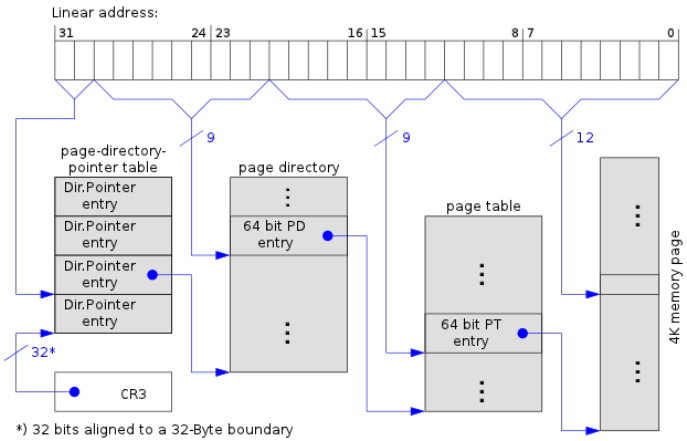
Modelo idealizado

- Direcciones: ...
- Mappings ...
- Páginas ...

En la realidad

- Tablas de páginas multi-level
- Cache y TLB
- Dispositivos (I/O)
- ...





Estados

$$
\begin{array}{lll}
State \stackrel{\text{def}}{=} \{ & active\_os & : os\_ident, \\
 & aos\_exec\_mode & : exec\_mode, \\
 & aos\_activity & : os\_activity, \\
 & oss & : os\_ident \mapsto os\_info, \\
 & hypervisor & : os\_ident \mapsto (padd \mapsto madd), \\
 & memory & : madd \mapsto page\}
\end{array}
$$

$$
os\_info \stackrel{\text{def}}{=} \{curr\_page : padd, hcall : option\ Hyper\_call\}
$$

Estados

$$
\begin{array}{lll}
State \stackrel{\text{def}}{=} \{ & active\_os & : os\_ident, \\
 & aos\_exec\_mode & : exec\_mode, \\
 & aos\_activity & : os\_activity, \\
 & oss & : os\_ident \mapsto os\_info, \\
 & hypervisor & : os\_ident \mapsto (padd \mapsto madd), \\
 & memory & : madd \mapsto page\}
\end{array}
$$

$$
os\_info \stackrel{\text{def}}{=} \{curr\_page : padd, hcall : option\ Hyper\_call\}
$$

Estado válido

Estados

$$
\begin{array}{lll}
State \stackrel{\text{def}}{=} \{ & active\_os & : os\_ident, \\
 & aos\_exec\_mode & : exec\_mode, \\
 & aos\_activity & : os\_activity, \\
 & oss & : os\_ident \mapsto os\_info, \\
 & hypervisor & : os\_ident \mapsto (padd \mapsto madd), \\
 & memory & : madd \mapsto page\}
\end{array}
$$

$$os\_info \stackrel{\text{def}}{=} \{curr\_page : padd, hcall : option\ Hyper\_call\}$$

Estado válido

- Muchas condiciones (ver el artículo), por ejemplo: toda dirección de máquina *ma* asociada a una dirección virtual en una tabla de páginas tiene una correspondiente pre-imagen, que es una dirección física, en el mapping del hipervisor

Estados

$$
\begin{array}{lll}
State \stackrel{\text{def}}{=} \{ & active\_os & : os\_ident, \\
 & aos\_exec\_mode & : exec\_mode, \\
 & aos\_activity & : os\_activity, \\
 & oss & : os\_ident \mapsto os\_info, \\
 & hypervisor & : os\_ident \mapsto (padd \mapsto madd), \\
 & memory & : madd \mapsto page\}
\end{array}
$$

$$os\_info \stackrel{\text{def}}{=} \{curr\_page : padd, hcall : option\ Hyper\_call\}$$

Estado válido

- Muchas condiciones (ver el artículo), por ejemplo: toda dirección de máquina *ma* asociada a una dirección virtual en una tabla de páginas tiene una correspondiente pre-imagen, que es una dirección física, en el mapping del hipervisor
- Invariante bajo ejecución: prueba larga y tediosa

Estados

$$
\begin{array}{lll}
State \stackrel{\text{def}}{=} \{ & active\_os & : os\_ident, \\
 & aos\_exec\_mode & : exec\_mode, \\
 & aos\_activity & : os\_activity, \\
 & oss & : os\_ident \mapsto os\_info, \\
 & hypervisor & : os\_ident \mapsto (padd \mapsto madd), \\
 & memory & : madd \mapsto page\}
\end{array}
$$

$$
os\_info \stackrel{\text{def}}{=} \{curr\_page : padd, hcall : option\ Hyper\_call\}
$$

Estado válido

- Muchas condiciones (ver el artículo), por ejemplo: toda dirección de máquina *ma* asociada a una dirección virtual en una tabla de páginas tiene una correspondiente pre-imagen, que es una dirección física, en el mapping del hipervisor
- Invariante bajo ejecución: prueba larga y tediosa
- Clave para la obtención de resultados de aislamiento y disponibilidad

(Algunas) Acciones

`read` *va*	Guest OS reads virtual address *va*.
`write` *va val*	Guest OS writes value *val* in *va*.
`new` *o va pa*	Hypervisor extends *os* memory with $va \mapsto ma$.
`del` *o va*	Hypervisor deletes mapping for *va* from current memory mapping of *o*.
`switch` *o*	Hypervisor sets *o* to be the active OS.
`hcall` *c*	Untrusted OS requires privileged service *c* to hypervisor.
`ret_ctrl`	Returns control to hypervisor.
`chmod`	Hypervisor changes execution mode from supervisor to user mode, and gives control to active OS.
`page_pin` *o pa t*	Registers memory page of type *t* at address *pa*.
`page_unpin` *o pa*	Memory page at *pa* is un-registered.

Semántica

- Pre-condición $Pre : State \rightarrow Action \rightarrow Prop$
- Post-condición $Post : State \rightarrow Action \rightarrow State \rightarrow Prop$
- Foco en ejecuciones normales: no se provee semantica para casos de error

Semántica de la acción `write`

$$Pre\ s\ (\texttt{write}\ va\ val) \stackrel{\text{def}}{=} os\_accessible(va) \wedge s.aos\_activity = running \wedge \exists\, ma : madd,\ va\_mapped\_to\_ma(s, va, ma) \wedge is\_RW((s.memory[ma]).page\_content)$$

$$Post\ s\ (\texttt{write}\ va\ val)\ s' \stackrel{\text{def}}{=} \exists\, ma : madd,\ va\_mapped\_to\_ma(s, va, ma) \wedge s'.memory = (s.memory[ma] := \langle RW(Some\ val), s.active\_os\rangle) \wedge s \sim_{memory,ma} s'$$

Propiedades de aislamiento

- **Read isolation**: ningún SO puede leer memoria que no le pertenece

Propiedades de aislamiento

- **Read isolation**: ningún SO puede leer memoria que no le pertenece
- **Write isolation**: un SO no puede modificar memoria que no le pertenece

Propiedades de aislamiento

- **Read isolation**: ningún SO puede leer memoria que no le pertenece
- **Write isolation**: un SO no puede modificar memoria que no le pertenece
- **OS isolation**: el comportamiento de un SO no depende del de otros SO guests

Propiedades de aislamiento

- **Read isolation**: ningún SO puede leer memoria que no le pertenece
- **Write isolation**: un SO no puede modificar memoria que no le pertenece
- **OS isolation**: el comportamiento de un SO no depende del de otros SO guests

Step-consistent unwinding lemma

$$\forall\ (s_1\ s_1'\ s_2\ s_2' : State)\ (a : Action)\ (osi : os\_ident),$$
$$s_1 \equiv_{osi} s_2 \rightarrow s_1 \xhookrightarrow{a} s_1' \rightarrow s_2 \xhookrightarrow{a} s_2' \rightarrow s_1' \equiv_{osi} s_2'$$

Propiedades de aislamiento

- **Read isolation**: ningún SO puede leer memoria que no le pertenece
- **Write isolation**: un SO no puede modificar memoria que no le pertenece
- **OS isolation**: el comportamiento de un SO no depende del de otros SO guests

Step-consistent unwinding lemma

$$\forall\ (s_1\ s_1'\ s_2\ s_2' : State)\ (a : Action)\ (osi : os\_ident),$$
$$s_1 \equiv_{osi} s_2 \rightarrow s_1 \xrightarrow{a} s_1' \rightarrow s_2 \xrightarrow{a} s_2' \rightarrow s_1' \equiv_{osi} s_2'$$

Locally preserves unwinding lemma

$$\forall\ (s\ s' : State)\ (a : Action)\ (osi : os\_ident),$$
$$\neg\ os\_action(s, a, osi) \rightarrow s \xrightarrow{a} s' \rightarrow s \equiv_{osi} s'$$

Propiedades de aislamiento

- **Read isolation**: ningún SO puede leer memoria que no le pertenece
- **Write isolation**: un SO no puede modificar memoria que no le pertenece
- **OS isolation**: el comportamiento de un SO no depende del de otros SO guests

Step-consistent unwinding lemma

$$\forall\ (s_1\ s_1'\ s_2\ s_2' : State)\ (a : Action)\ (osi : os\_ident),$$
$$s_1 \equiv_{osi} s_2 \rightarrow s_1 \xrightarrow{a} s_1' \rightarrow s_2 \xrightarrow{a} s_2' \rightarrow s_1' \equiv_{osi} s_2'$$

Locally preserves unwinding lemma

$$\forall\ (s\ s' : State)\ (a : Action)\ (osi : os\_ident),$$
$$\neg\ os\_action(s, a, osi) \rightarrow s \xrightarrow{a} s' \rightarrow s \equiv_{osi} s'$$

Non-influence on traces

$$\forall\ (ss_1\ ss_2 : Trace)\ (osi : os\_ident),$$
$$(ss_1[0] \equiv_{osi} ss_2[0]) \rightarrow same\_act(osi, ss_1, ss_2) \rightarrow \Box(\equiv_{osi}, ss_1, ss_2)$$

Disponibilidad

- SI el hipervisor solamente ejecuta acciones `chmod` cuando no hay *hypercall* pendientes
- Y el hipervisor retorna, infinitamente a menudo, el control a los SO
- ENTONCES ningún SO se bloquea indefinidamente esperando que sus hypercalls sean atendidas

$$\forall\,(ss : Trace), \neg\, hcall(ss[0]) \rightarrow \Box(chmod\_nohcall, ss) \rightarrow \Box(\Diamond \neg\, hyper\_running, ss) \rightarrow \Box(\Diamond \neg\, hcall, ss)$$

Fairness y otras propiedades

- No garantiza que todo SO sea finalmente atendido
- Otras políticas pueden ser consideradas

Estadísticas

Modelo y lemas básicos	4.8	kLOC
Invarianza de estado válido	8.0	kLOC
Read y write isolation	0.6	kLOC
OS isolation	6.0	kLOC
Disponibilidad	1.0	kLOC
Total	**20.4**	**kLOC**

Estadísticas

Modelo y lemas básicos	4.8	kLOC
Invarianza de estado válido	8.0	kLOC
Read y write isolation	0.6	kLOC
OS isolation	6.0	kLOC
Disponibilidad	1.0	kLOC
Total	**20.4**	**kLOC**

Código Coq

Disponible en `http://www.fing.edu.uy/inco/grupos/gsi/proyectos/virtualcert.php`

Contribuciones

- Modelo idealizado de virtualización formalizado
- Pruebas de aislamiento y disponibilidad verificadas (machine checked)

Contribuciones

- Modelo idealizado de virtualización formalizado
- Pruebas de aislamiento y disponibilidad verificadas (machine checked)

Trabajo futuro

- Completar el modelo:
  - Cache y TLB
    (completadas para políticas *write through* y *total flushing*)
  - Dispositivos
- Prueba de nuevas propiedades:
  - *indistinguishability* (completada)
  - más políticas de disponibilidad
- Verificación de una implementación en C usando *relational Hoare logics* (comenzado)

Referencias


G. Barthe, G. Betarte, J.D. Campo, C. Luna.
Formally verifying isolation and availability in an idealized model of virtualization.
Proceedings of FM2011: 17th International Symposium on Formal Methods, Lecture Notes in Computer Science, vol. 6664, pp 231-245, June 2011.


Página del GSI
`http://www.fing.edu.uy/inco/grupos/gsi`

Muchas gracias!!

Preguntas?