Estudio comparativo de frameworks de Instrumentación Dinámica de Ejecutables

Proyecto Fin de Carrera de Ingeniería en Informática

Juan Antonio Artal Lozano

Director: Ricardo J. Rodríguez Fernández
Ponente: José J. Merseguer Hernáiz

10 de Mayo de 2012

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Contenidos

Universidad Zaragoza
1542

Contenidos

Universidad Zaragoza 1542

Introducción (I): Instrumentación

¿Qué es?

Inserción de código adicional

- **Antes** de compilar, código fuente
- **Después** de compilar, ejecutable

¿Qué permite?

Recogida de información en tiempo de ejecución

- Estudio de arquitecturas
  - Modelado de cachés
  - Simulación de nuevas arquitecturas
- Análisis de código
  - Rendimiento

Zaragoza 1542

Introducción (II): Instrumentación

Tipos

- Manual
- *Automated source level*
- Asistida por el compilador
- *Binary translation*
- DBI (*Dynamic binary instrumentation*)

Introducción (II): Instrumentación

Tipos

- Manual
- *Automated source level*
- Asistida por el compilador
- *Binary translation*
- DBI (*Dynamic binary instrumentation*)
  - Herramienta añade código a un ejecutable en tiempo de ejecución

Universidad Zaragoza 1542

Introducción (II): Instrumentación

Tipos

- Manual
- *Automated source level*
- Asistida por el compilador
- *Binary translation*
- DBI (*Dynamic binary instrumentation*)
  - Herramienta añade código a un ejecutable en tiempo de ejecución

DBI

- Más completa y moderna
- Monitoriza y controla desde la 1ª instrucción

Universidad Zaragoza 1542

Introducción (III)

Framework de DBI

- Permite programar aplicaciones para instrumentar
- API $\Rightarrow$ herramamienta de análisis dinámico (DBA)
- Ejemplos: Pin, Valgrind, DynamoRIO...

Desventajas de DBI

- Sobrecarga al instrumentar en tiempo de ejecución
- $\Downarrow$ rendimiento
- Muy pocas comparativas a nivel de rendimiento
- $\nexists$ comparativas a nivel de programación

Universidad Zaragoza

Introducción (IV): Objetivo del PFC

Estudio comparativo a nivel de impacto en rendimiento

Hitos intermedios:

- Estudio de alternativas de frameworks de DBI
- Desarrollo de herramientas DBA
- Creación de un benchmark propio
- Experimentos

Universidad Zaragoza 1542

Contenidos

Universidad Zaragoza

Conocimientos previos (I): Frameworks de DBI

¿Qué es?

APIs, librerías y software de instrumentación.

Conocimientos previos (I): Frameworks de DBI

¿Qué es?

APIs, librerías y software de instrumentación.

Componentes principales

- Núcleo: compilador *just-in-time* (JIT)
- Herramientas (plugins o librerías):
  - Instrumentación: dónde y qué
  - Análisis: ejecución

Universidad Zaragoza 1542

Conocimientos previos (I): Frameworks de DBI

¿Qué es?

APIs, librerías y software de instrumentación.

Componentes principales

- Núcleo: compilador *just-in-time* (JIT)
- Herramientas (plugins o librerías):
  - Instrumentación: dónde y qué
  - Análisis: ejecución

Granularidad

Instrucción	Traza
Bloque básico	Rutina
Superbloque	Imagen

1542

Conocimientos previos (II)

Origen de DBI

- Primeras Herramientas
  - Pixie [SG92], Epoxie [Wal91] y QPT [LB94]
- Simuladores
  - Tango Lite [GH93], Proteus [BDCW91] y g88 [Bed90]
- Primer framework de DBI
  - ATOM [SE94], API de instrumentación para recompilar la aplicación

Universidad Zaragoza 1542

Contenidos

Universidad Zaragoza
1542

Framework de DBI (I): Alternativas y criterios

Alternativas estudiadas

Pin	DynInst	
Valgrind	Dtrace	HDtrans
DynamoRIO	Systemtap	

Criterios de selección

- Software en desarrollo
- Licencia con acceso al fuente
- Gratuito
- API de desarrollo
- S.O. y arquitectura común

Framework de DBI (I): Alternativas y criterios

Alternativas estudiadas

✓Pin	DynInst	
✓Valgrind	Dtrace	HDtrans
✓DynamoRIO	Systemtap	

Criterios de selección

- Software en desarrollo
- Licencia con acceso al fuente
- Gratuito
- API de desarrollo
- S.O. y arquitectura común

Frameworks de DBI (II): Seleccionados

Características

Código fuente disponible (GNU GPL v2)
Representación intermedia VEX IR
Herramientas pesadas

Framework	Versión	Arquitecturas	S.O.	Granularidad
Valgrind	3.7.0 05/11/2011	Arm, PowerPC, s390, x86, x64	Android, OSX, Linux	I S

Granularidades: Instrucciones Bloque básico Superbloque Traza Rutina IMagen

Universidad Zaragoza 1542

Frameworks de DBI (II): Seleccionados

Características

Intel
Código fuente disponible (Licencia propietaria)
Vinculación a proceso en ejecucion

Framework	Versión	Arquitecturas	S.O.	Granularidad
Valgrind	3.7.0 05/11/2011	Arm, PowerPC, s390, x86, x64	Android, OSX, Linux	I S
Pin	2.10 28/11/2011	Arm, IA-64, x86, x64	Windows, Linux	I B T R M

Granularidades: Instrucciones Bloque básico Superbloque Traza Rutina IMagen

Universidad Zaragoza
1542

Frameworks de DBI (II): Seleccionados

Características

MIT, HP, Google
Código fuente disponible (BSD-2)
Bien documentado

Framework	Versión	Arquitecturas	S.O.	Granularidad
Valgrind	3.7.0 05/11/2011	Arm, PowerPC, s390, x86, x64	Android, OSX, Linux	I S
Pin	2.10 28/11/2011	Arm, IA-64, x86, x64	Windows, Linux	I B T R M
DynamoRIO	2.2.0 01/04/2011	x86, x64	Windows, Linux	I B T

Granularidades: Instrucciones Bloque básico Superbloque Traza Rutina IMagen

Universidad Zaragoza

Frameworks de DBI (II): Seleccionados

Características

MIT, HP, Google
Código fuente disponible (BSD-2)
Bien documentado

Framework	Versión	Arquitecturas	S.O.	Granularidad
Valgrind	3.7.0 05/11/2011	Arm, PowerPC, s390, x86, x64	Android, OSX, Linux	I S
Pin	2.10 28/11/2011	Arm, IA-64, x86, x64	Windows, Linux	I B T R M
DynamoRIO	2.2.0 01/04/2011	x86, x64	Windows, Linux	I B T

Granularidades: Instrucciones Bloque básico Superbloque Traza Rutina IMagen

Similitudes

- Código inyectado en C/C++
- No es necesario codigo fuente de la aplicación a instrumentar

Frameworks de DBI (II): Seleccionados

Características

MIT, HP, Google
Código fuente disponible (BSD-2)
Bien documentado

Framework	Versión	Arquitecturas	S.O.	Granularidad
Valgrind	3.7.0 05/11/2011	Arm, PowerPC, s390, x86, x64	Android, OSX, Linux	I S
Pin	2.10 28/11/2011	Arm, IA-64, x86, x64	Windows, Linux	I B T R M
DynamoRIO	2.2.0 01/04/2011	x86, x64	Windows, Linux	I B T

Granularidades: Instrucciones Bloque básico Superbloque Traza Rutina IMagen

Similitudes

- Código inyectado en C/C++
- No es necesario codigo fuente de la aplicación a instrumentar
- GNU/Linux x86

Contenidos

Universidad Zaragoza 1542

Trabajo relacionado (I)

Trabajos relacionados

Chen et al. [CKS+08]
Método para disminuir sobrecarga en granularidad de instrucciones

Trabajo	Frameworks	Instrumentación	Benchmarks
[CKS+08]	Pin, Valgrind	Instrucciones	SPEC 2000

Trabajo relacionado (I)

Trabajos relacionados

Guha et al. [Sof07]
Pruebas de rendimiento de Strata

Trabajo	Frameworks	Instrumentación	Benchmarks
[CKS+08]	Pin, Valgrind	Instrucciones	SPEC 2000
[Sof07]	Pin, Valgrind, DynamoRIO	Bloques	SPEC 2000

Universidad Zaragoza

Trabajo relacionado (I)

Trabajos relacionados

Bellard [Bel05]
Pruebas de rendimiento de Qemu

Trabajo	Frameworks	Instrumentación	Benchmarks
[CKS+08]	Pin, Valgrind	Instrucciones	SPEC 2000
[Sof07]	Pin, Valgrind, DynamoRIO	Bloques	SPEC 2000
[Bel05]	Qemu, Bochs, Valgrind	Bloques	BYTEmark

Universidad Zaragoza

Trabajo relacionado (I)

Trabajos relacionados

Ruiz et al. [RAH08]
Pruebas de rendimiento con procesadores de diferentes cachés

Trabajo	Frameworks	Instrumentación	Benchmarks
[CKS+08]	Pin, Valgrind	Instrucciones	SPEC 2000
[Sof07]	Pin, Valgrind, DynamoRIO	Bloques	SPEC 2000
[Bel05]	Qemu, Bochs, Valgrind	Bloques	BYTEmark
[RAH08]	Pin, DynamoRIO	*X*	SPEC 2006

Universidad Zaragoza

Trabajo relacionado (I)

Trabajos relacionados

Weaver et al. [WM08]
Pruebas de rendimiento de herramientas DBA

Trabajo	Frameworks	Instrumentación	Benchmarks
[CKS+08]	Pin, Valgrind	Instrucciones	SPEC 2000
[Sof07]	Pin, Valgrind, DynamoRIO	Bloques	SPEC 2000
[Bel05]	Qemu, Bochs, Valgrind	Bloques	BYTEmark
[RAH08]	Pin, DynamoRIO	*X*	SPEC 2006
[WM08]	Pin, Valgrind, Qemu	Bloques	SPEC 2000, 2006

Universidad Zaragoza 1542

Trabajo relacionado (I)

Trabajos relacionados

Uh et al. [PA06]
Método para averiguar sobrecarga en frameworks de DBI

Trabajo	Frameworks	Instrumentación	Benchmarks
[CKS+08]	Pin, Valgrind	Instrucciones	SPEC 2000
[Sof07]	Pin, Valgrind, DynamoRIO	Bloques	SPEC 2000
[Bel05]	Qemu, Bochs, Valgrind	Bloques	BYTEmark
[RAH08]	Pin, DynamoRIO	*X*	SPEC 2006
[WM08]	Pin, Valgrind, Qemu	Bloques	SPEC 2000, 2006
[PA06]	Pin	Instr., Bloques	SPEC 2000

Universidad Zaragoza

Trabajo relacionado (II): Contribución

Contribución

- Instrumentación por instrucciones y bloques
- Benchmark específico
- Información sobre:
  - Instrucciones y bloques
  - Consumo de memoria

Contenidos

Universidad Zaragoza 1542

Creación de benchmark (I): Alternativas

Alternativas estudiadas

PCMark	TPC
3DMark	Parsec
Sysmark 2012	SPEC CPU

Creación de benchmark (I): Alternativas

Alternativas estudiadas

PCMark	TPC
3DMark	Parsec
Sysmark 2012	SPEC CPU

SPEC CPU

- ✓ Benchmark para múltiples S.O.
- ✓ Permite integrar la llamada al framework de DBI
- ✓ Buena metodología

Creación de benchmark (I): Alternativas

Alternativas estudiadas

PCMark	TPC
3DMark	Parsec
Sysmark 2012	SPEC CPU

SPEC CPU

- ✓ Benchmark para múltiples S.O.
- ✓ Permite integrar la llamada al framework de DBI
- ✓ Buena metodología
- X Compara tu máquina con una SUN de 1997
- X No ofrece información sobre el consumo de memoria
- X De pago

Creación de benchmark (II): Requisitos y definición

Requisitos [Cor06]

- Carga de trabajo
- Métrica
  - Velocidad
  - *Throughput*

Definición

- Métodos similares a los de SPEC
- Reutilización de 9 aplicaciones de SPEC
- Generación información de tiempo y memoria usados
- Información de instrucciones y bloques básicos

Creación de benchmark (II): Requisitos y definición

Requisitos [Cor06]

- Carga de trabajo
- Métrica
  - Velocidad
  - *Throughput*

Definición

- Métodos similares a los de SPEC
- Reutilización de 9 aplicaciones de SPEC
- Generación información de tiempo y memoria usados
- Información de instrucciones y bloques básicos
- ✓ Benchmark más rápido que SPEC

Creación de benchmark (III): Aplicaciones seleccionadas

Cálculo entero

- `bzip2`
- `GNU go`
- `hmmer`
- `h264ref`
- `libquantum`

Creación de benchmark (III): Aplicaciones seleccionadas

Cálculo entero

- `bzip2`
- `GNU go`
- `hmmer`
- `h264ref`
- `libquantum`

Cálculo real

- `namd`
- `povray`
- `milc`
- `mlucas`
- `linpack`

Universidad Zaragoza 1542

Creación de benchmark (III): Aplicaciones seleccionadas

Cálculo entero

- `bzip2`
- `GNU go`
- `hmmer`
- `h264ref`
- `libquantum`

Cálculo real

- `namd`
- `povray`
- `milc`
- `mlucas`
- `linpack`

E/S

- `whirlpool`
- `ripemd`
- `aes`
- `ffmpeg`

Universidad Zaragoza 1542

Creación de benchmark (III): Aplicaciones seleccionadas

Cálculo entero

- bzip2
- GNU go
- hmmer
- h264ref
- libquantum

Cálculo real

- namd
- povray
- milc
- mlucas
- linpack

E/S

- whirlpool
- ripemd
- aes
- ffmpeg

Memoria

- memtester

Universidad Zaragoza
1542

Creación de Benchmark (IV): Herramientas DBA programadas

Objetivo del benchmark

Evaluación del rendimiento de los frameworks de DBI con las herramientas DBA programadas

Slowdown

Relación entre el tiempo de ejecución instrumentado y de forma nativa, sin instrumentar.

Aprendizaje de las APIs

- Pin: $\uparrow$ Documentación, $\uparrow\uparrow$ Ejemplos, $\uparrow$ Tutoriales
- DynamoRIO: $\uparrow\uparrow$ Documentación, $\uparrow$ Ejemplos, $\uparrow$ Tutoriales
- Valgrind: $\downarrow$ Documentación, $\downarrow$ Ejemplos, $\downarrow$ Tutoriales

Creación de Benchmark (V): Herramientas DBA programadas

Herramientas DBA

Programadas para instrumentar a diferente granularidad

A nivel de instrucción

Generación del mínimo *slowdown* máximo.

Total **instrucciones** ejecutadas

- pin.instrucciones
- valgrind.instrucciones
- dynamorio.instrucciones

A nivel de bloque básico

Generación de *slowdown* medio.

Total **bloques** ejecutados

- pin.bloques
- valgrind.bloques
- dynamorio.bloques

d

Creación de benchmark (VI): Algoritmo del benchmark

1: $optimizacion = [O0, O3]$
2: $framework = [Pin, Valgrind, DynamoRIO]$
3: $aplicacion = [bzip2, go, hmmer, h264ref, libquantum, namd, povray...]$
4: **for** *aplicacion* **do**
5: **for** *optimizacion* **do**
6: **for** *repeticion* = 1 to 3 **do**
7: ejecutar *aplicacion.optimizacion* {Ejecución nativa}
8: **for** *framework* **do**
9: ejecutar *framework*.**instrucciones** *aplicacion.optimizacion*
10: **end for**
11: **for** *framework* **do**
12: ejecutar *framework*.**bloques** *aplicacion.optimizacion*
13: **end for**
14: **end for**
15: **end for**
16: **end for**

Universidad Zaragoza 1542

Contenidos

Universidad Zaragoza 1542

Experimentos (I): Entorno de pruebas

Hardware

Nombre CPU	Intel®Core™2 Duo CPU T7300
Características	2.00 GHz, 667 MHz bus
CPU(s)	2 cores, 1 desactivado.
Caché primer nivel	32 KiB I + 32 KiB D por core
Caché segundo nivel	4 MiB I+D
Memoria RAM	2 GiB (2x1 GiB DDR2 SODIMM 667 Mhz)
Disco Duro	120 GB HITACHI HTS54161

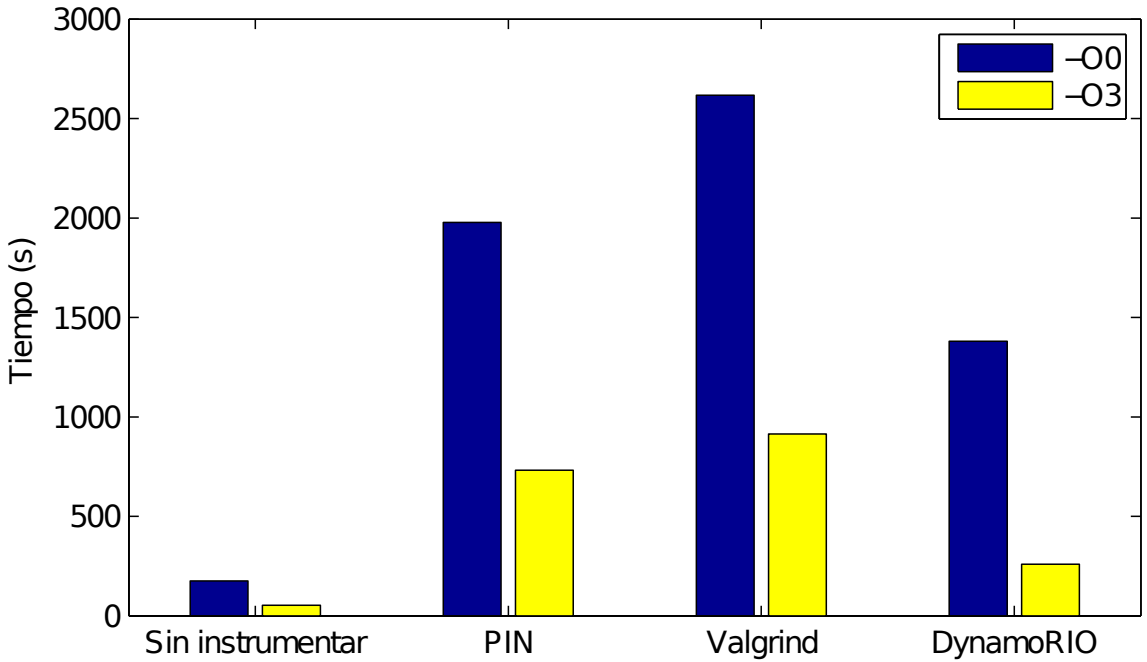
Software

Sistema Operativo	Fedora Core 14 32bit
Compilador C	gcc (GCC) 4.5.1 20100924 (Red Hat 4.5.1-4)
Compilador Fortran	GNU Fortran 4.5.1
Sistema de ficheros	ext4
Nivel sistema	Run level 3 (multiusuario)

rsidad
za

Experimentos (II): Resultados

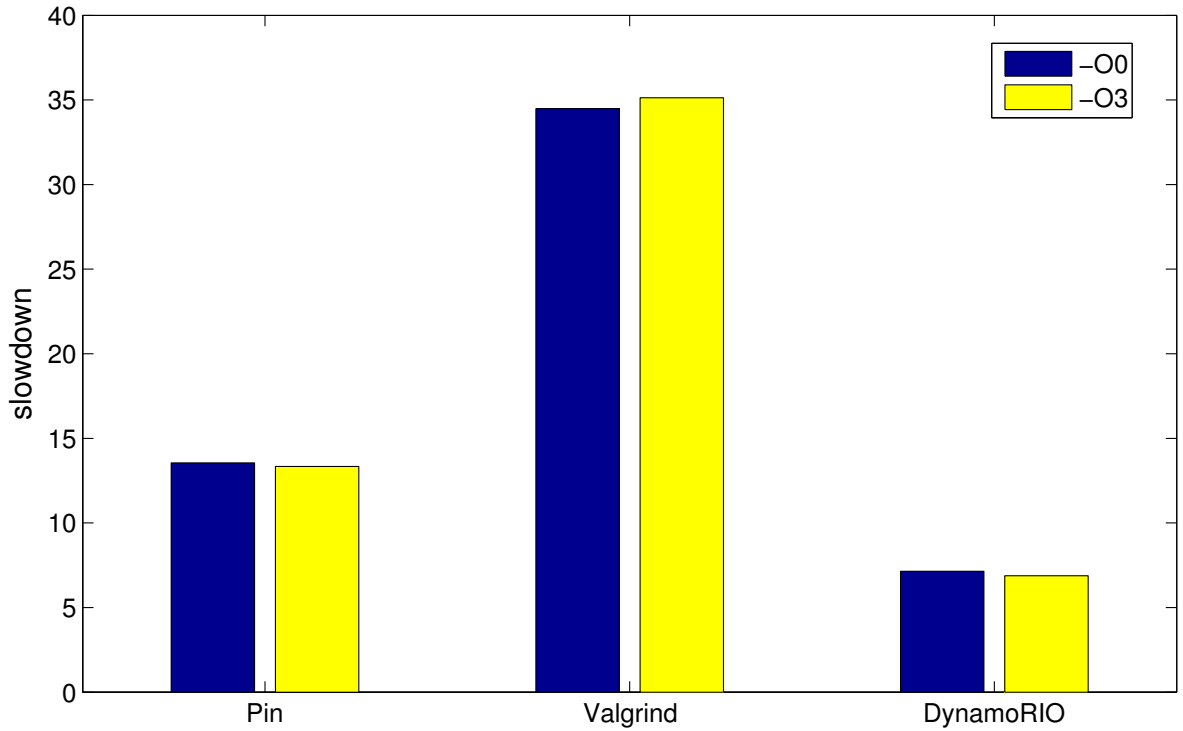
Tiempo de ejecución `h264ref`



Universidad Zaragoza
1542

Experimentos (III): Resultados

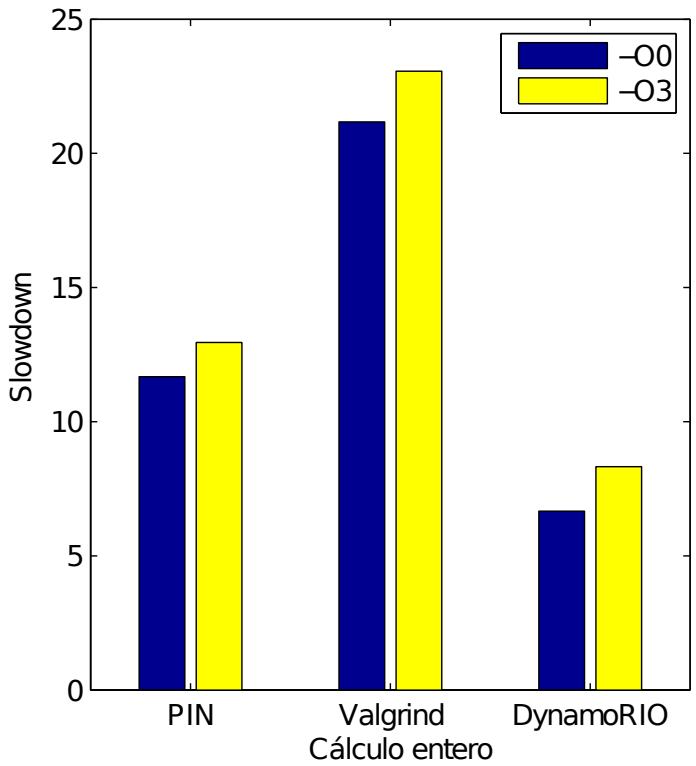
Slowdown en `ffmpeg`

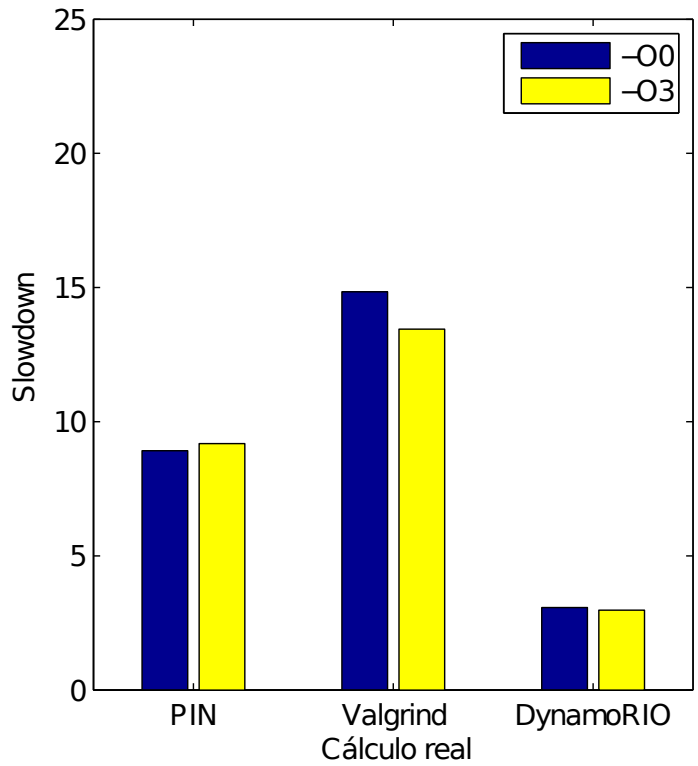


Universidad Zaragoza
1542

Experimentos (IV): Resultados

Slowdown medio benchmark instrucciones

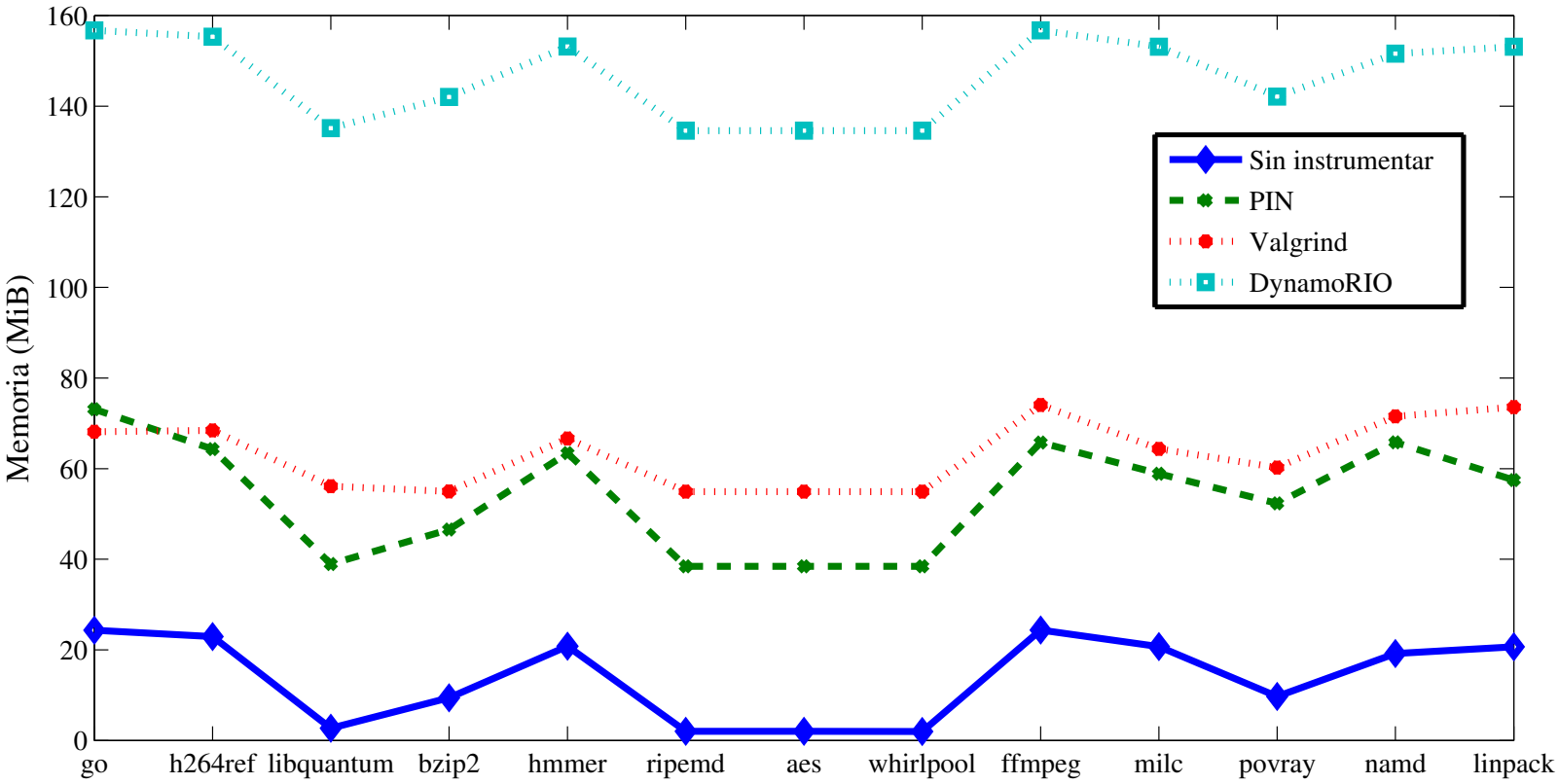




Universidad Zaragoza
1542

Experimentos (V): Resultados

Consumo medio de memoria



Experimentos (VI): Resultados

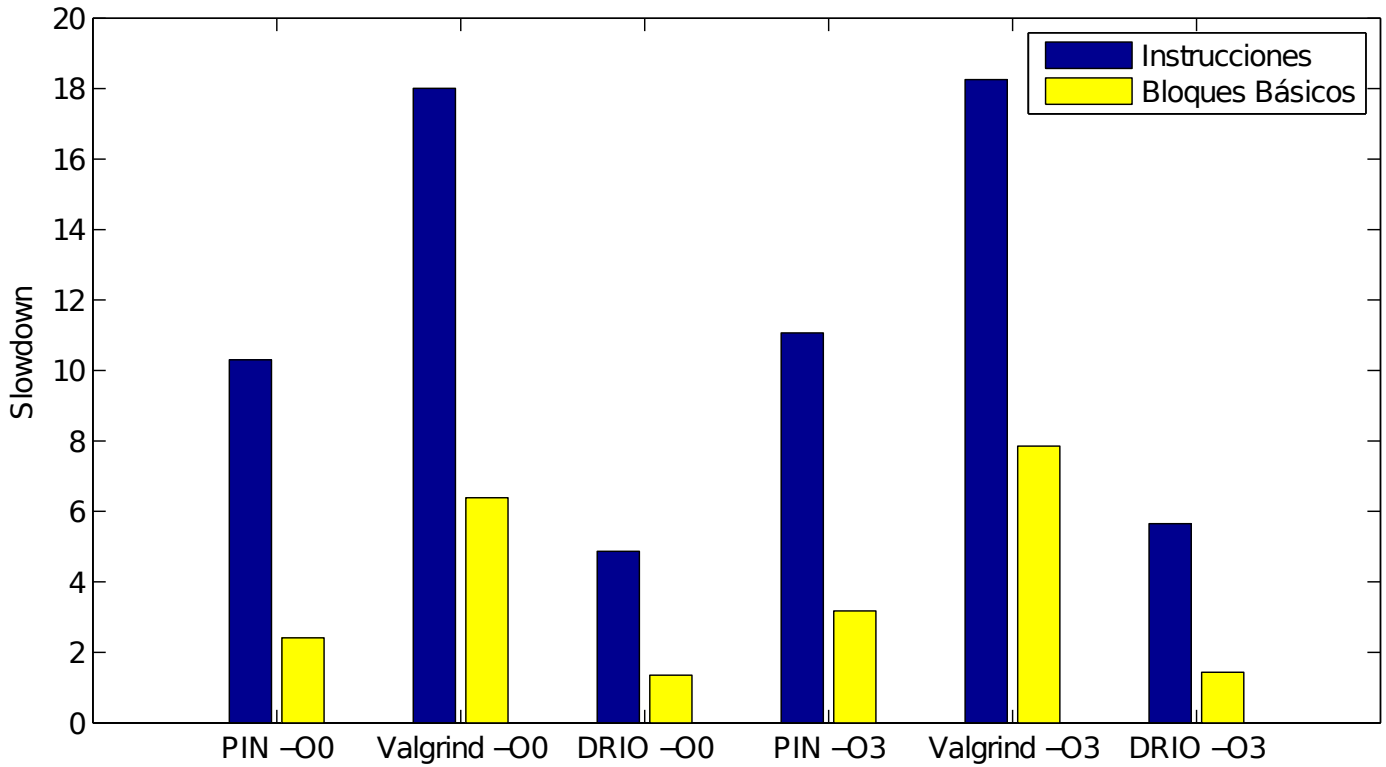
Consumo medio de memoria por Frameworks

Framework	Media consumo (kiB)	Desviación media (kiB)
Pin -O0	39885	3113
Valgrind -O0	53033	3736
DynamoRIO -O0	132478	73
Pin -O3	40871	4006
Valgrind -O3	52789	3535
DynamoRIO -O3	132475	72

Universidad Zaragoza
1542

Experimentos (VII): Resultados

Slowdown por instrumentaciones



Universidad Zaragoza
1542

Contenidos

Universidad Zaragoza 1542

Conclusiones y trabajo futuro (I): Conclusiones

Conclusiones PFC

- ✓ Nivel optimización o tipo cálculo → DynamoRIO
- X Más lenta → Valgrind
- Consumo de memoria
  - ✓ ↓ Pin
  - X ↑ DynamoRIO

Universidad Zaragoza 1542

Conclusiones y trabajo futuro (I): Conclusiones

Conclusiones PFC

- ✓ Nivel optimización o tipo cálculo → DynamoRIO
- X Más lenta → Valgrind
- Consumo de memoria
  - ✓ ↓ Pin
  - X ↑ DynamoRIO

Conclusiones personales

- Alcanzado los objetivos previamente definidos
- Ampliado el conocimiento sobre:
  - DBI
  - Depuración y análisis ejecutables

Universidad Zaragoza 1542

Conclusiones y trabajo futuro (II): Líneas futuras

Líneas futuras

- Ampliación frameworks de DBI
- Ampliación aplicaciones para benchmark
- Mediciones instrucciones con Linux Performance-Monitoring Driver
- Cambio medición tiempo e instrucciones con Intel VTune Amplifier XE
- Profundizar en las APIs de los frameworks de DBI

Universidad Zaragoza 1542

Referencias (I)

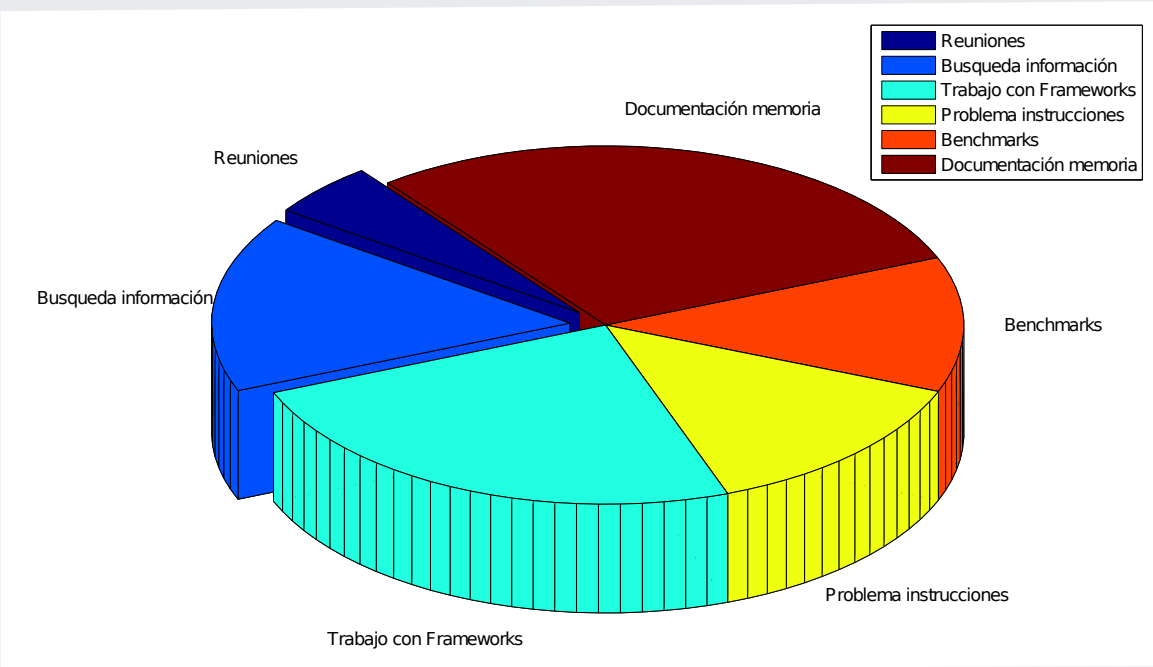
- **[BDCW91]** Eric A. Brewer, Chrysanthos N. Dellarocas, Adrian Colbrook, and William E. Weihl. **PROTEUS: A High-Performance Parallel-Architecture Simulator**. *Technical report, 1991.*
- **[Bed90]** Robert Bedichek. **Some efficient architectures simulation techniques**. In *Winter 1990 USENIX Conference.*
- **[Bel05]** Fabrice Bellard. **QEMU, a fast and portable dynamic translator**. In *Proceedings of the annual conference on USENIX Annual Technical Conference, ATEC '05*, pages 41-41, Berkeley, CA, USA, 2005. USENIX Association.
- **[CKS+08]** Shimin Chen, Michael Kozuch, Theodoros Strigkos, Babak Falsafi, Phillip B.Gibbons, Todd C. Mowry, Vijaya Ramachandran, Olatunji Ruwase, Michael Ryan, and Evangelos Vlachos. **Flexible hardware acceleration for instructiongrain program monitoring**. *SIGARCH Comput. Archit. News*, 36(3):377-388, jun 2008.
- **[Cor06]** Standard Performance Evaluation Corporation. **SPECCPU 2006 benchmarks**. http://www.spec.org, 2006.
- **[GH93]** Stephen R. Goldschmidt and John L. Hennessy. **The accuracy of trace-driven simulations of multiprocessors**. In *Proceedings of the 1993 ACM SIGMETRICS conference on Measurement and modeling of computer systems, SIGMETRICS '93*, pages 146-157, New York, NY, USA, 1993. ACM.
- **[LB94]** James R. Larus and Thomas Ball. **Rewriting executable files to measure program behavior**. *Softw. Pract. Exper., 24(2)*:197-218, February 1994.

Referencias (II)

- **[PA06]** G.-R. Uh; R. Cohn; B. Yadavalli; R. Peri and R. Ayyagari. **Analyzing dynamic binary instrumentation overhead**. In *Workshop on Binary Instrumentation and Applications, WBIA, 2006.*
- **[RAH08]** Arkaitz Ruiz-Alvarez and Kim M. Hazelwood. **Evaluating the impact of dynamic binary translation systems on hardware cache performance**. In *IISWC*, pages 131-140, 2008.
- **[SE94]** Amitabh Srivastava and Alan Eustace. **ATOM: a system for building customized program analysis tools**. *SIGPLAN Not.*, 29(6):196-205, June 1994.
- **[SG92]** Inc. Silicon Graphics. **MIPS Assembly Language Programmer's Guide**. Silicon Graphics, Inc., 1992.
- **[Sof07]** A. Guha; J.D. Hiser; Naveen Kumar; J. Yang; M. Zhao; S. Zhou; B.R. Childers; J.W. Davidson; K. Hazelwood; M.L. Soffa. **Virtual execution environments: Support and tools**. In *International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Dept. of Comput. Sci., Virginia Univ., Charlottesville, VA, March 2007.
- **[Wal91]** DavidW.Wall. **Systems for late code modification**. In *WRL Research Report 91/5*, pages 275-293. Springer-Verlag, 1991.
- **[WM08]** Vincent M. Weaver and Sally A. McKee. **Using dynamic binary instrumentation to generate multi-platform simpoints: methodology and accuracy**. In *Proceedings of the 3rd international conference on High performance embedded architectures and compilers, HiPEAC'08*, pages 305-319, Berlin, Heidelberg, 2008. Springer-Verlag.

Tiempo dedicado

Esfuerzo temporal



Tareas	Horas
Reuniones	20
Información	75
Frameworks	95
Problemas	55
Benchmark	60
Memoria	105
Total	**410**