Fragmentation Problem 000 00000000000000000000000 Compact-Fit

Partial Compaction

Experiments

Conclusion O

# A Compacting Real-Time Memory Management System

Silviu S. Craciunas, Christoph M. Kirsch, Hannes Payer, Ana Sokolova, Horst Stadler, Robert Staudinger

Hannes Payer

Computational Systems Group, University of Salzburg

July 1, 2008

▲□▷ 《团▷ 《필▷ 《필▷ 》를 
Computational Systems Group, University of Salzburg

Introduc	tion Fragmer	tation Problem	Compact-Fit	Partial Compaction	Experiments	Conclusion
00	000	000000000000000000000000000000000000000	000	00000000	000	0

#### Overview

#### Introduction

Compact-Fit

Experiments

Conclusion

Computational Systems Group, University of Salzburg

æ

ntroduction	Fragmentation
0	000
00	0000000000

nentation Problem

Compact-Fit

Partial Compaction

Experiment:

Conclusion O

# Motivation

Traditional dynamic memory management systems are typically non-deterministic:

- unpredictable response times of memory operations
- unpredictable memory fragmentation

⇒ Dynamic memory management systems are typically not used in time-critical software components (hard real-time systems, device drivers ...)

Fragmentation Problem 000 00000000000000000000000 Compact-Fit 000 Partial Compaction

Experiments

Conclusion 0

#### Predictable Memory Management System

#### Predictability in Time

The time a memory management operation takes is determined by the size of the object involved in the operation (allocation, deallocation, and dereference).

#### Predictability in Space

The number of actual allocations together with their sizes (not the order of invocations) determines how many more allocations of a given size will succeed before running out of memory.

Introduction	
00	
•0	

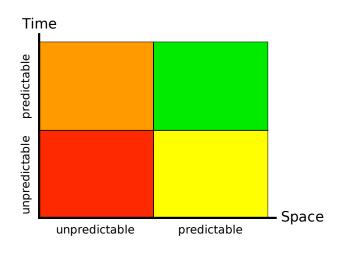
Fragmentation Problem

Compact-Fit 000 Partial Compaction

Experiments

Conclusion O

### Solution Space



æ

ヘロン ヘロン ヘビン ヘビン

Introduction	
00	
00	

Fragmentation Problem

Compact-Fit 000 Partial Compaction

Experiments

Conclusion O

# What We Want?

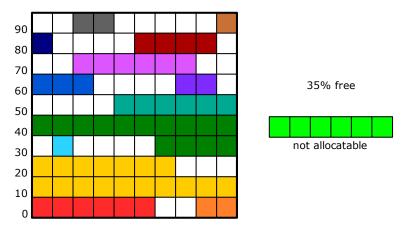
A memory management system predictable in time and space (component of the real-time operating system Tiptoe)

Properties:

- malloc(n) takes at most  $\mathcal{O}(n)$  time
- free(n) takes at most  $\mathcal{O}(n)$  time
- memory access (dereference) takes small constant time
- small and predictable memory fragmentation bound

Fragmentation Problem .00

#### Fragmentation Problem



fragmentation in a contiguous space  $\Rightarrow$ compaction  $\Rightarrow$  reference updates

> ・ロン ・回 と ・ ヨ と ・ ヨ と Computational Systems Group, University of Salzburg

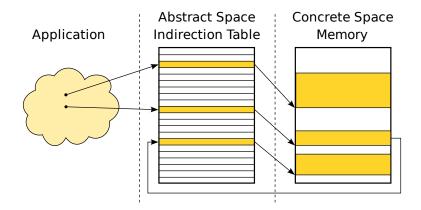
Fragmentation Problem

Compact-Fi 000 Partial Compaction

Experiments

Conclusion O

#### Solution to Reference Updates



◆□> ◆母> ◆量> ◆量> 量 のへの Computational Systems Group, University of Salzburg

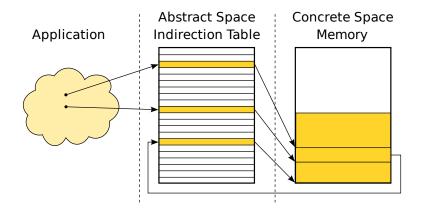
Fragmentation Problem

Compact-Fi 000 Partial Compaction

Experiments

Conclusion O

#### Solution to Reference Updates



・ロト・日本・モート・モート きつのの
Computational Systems Group, University of Salzburg

Fragmentation Problem

Compact-Fit

Partial Compaction

Experiments

Conclusion

# Compaction

Trade-Off:

speed versus memory fragmentation

#### Requirement:

keep speed and memory fragmentation bounded and predictable

#### 2 Extreme Non-Solutions:

- keep memory perfectly compact
- perform memory operations in constant time without considering memory fragmentation

Fragmentation Problem

Compact-Fit

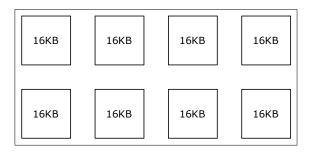
Partial Compaction

Experiments

Conclusion O

#### Concrete Address Space

• concrete address space is divided into pages of equal size



æ

イロト イヨト イヨト イヨト

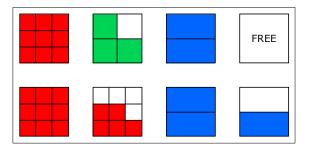
Introd	uction	
00		
00		

Partial Compaction

Experiments 000 Conclusion O

### Concrete Address Space

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks



Introd	uctio	n
00		
00		

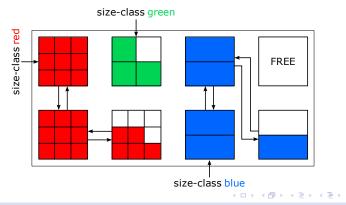
Partial Compaction

Experiments

Conclusion O

# Concrete Address Space

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks
- *n* predefined page-block sizes ⇒ *n* different **size-classes**



#### Computational Systems Group, University of Salzburg

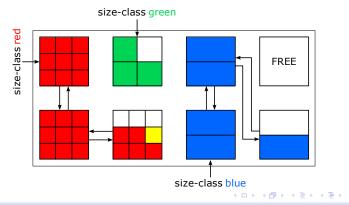
Introd	uction	
00		
00		

Partial Compaction

Experiments

Conclusion O

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks
- *n* predefined page-block sizes ⇒ *n* different **size-classes**



Computational Systems Group, University of Salzburg

Introd	uction
00	
00	

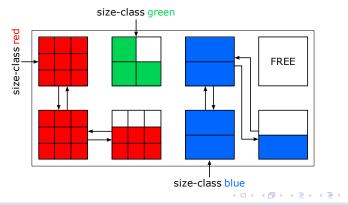
Partial Compaction

Experiments

Conclusion O

# Concrete Address Space

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks
- *n* predefined page-block sizes ⇒ *n* different **size-classes**



#### Computational Systems Group, University of Salzburg

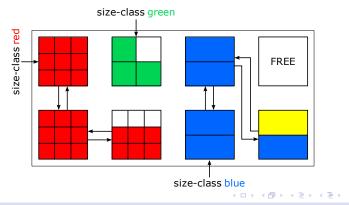
Introd	uction
00	
00	

Partial Compaction

Experiments

Conclusion O

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks
- *n* predefined page-block sizes ⇒ *n* different **size-classes**



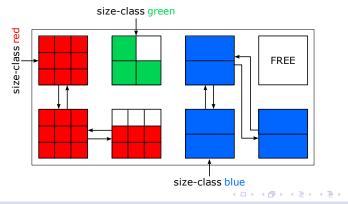
Computational Systems Group, University of Salzburg

Introd	uctio	n
00		
00		

Partial Compaction

Experiments 000 Conclusion O

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks
- *n* predefined page-block sizes ⇒ *n* different **size-classes**



Computational Systems Group, University of Salzburg

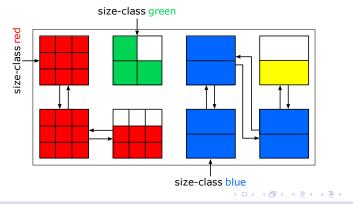
Introd	uction	
00		
00		

Partial Compaction

Experiments

Conclusion O

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks
- *n* predefined page-block sizes ⇒ *n* different **size-classes**



Computational Systems Group, University of Salzburg

Introd	uction	
00		
00		

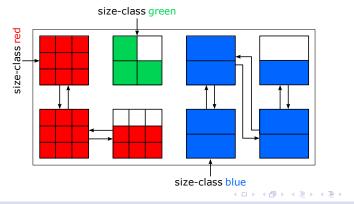
Partial Compaction

Experiments

Conclusion O

# Concrete Address Space

- concrete address space is divided into pages of equal size
- each page itself is divided into fixed-sized page-blocks
- *n* predefined page-block sizes ⇒ *n* different **size-classes**



#### Computational Systems Group, University of Salzburg

Fragmentation Problem

Compact-Fit 000 Partial Compaction

Experiments

Conclusion 0

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:

Each size-class can contain at most one not-full page.

・ロト・日ト・ヨト・ヨト ヨークへ
Computational Systems Group, University of Salzburg

Fragmentation Problem

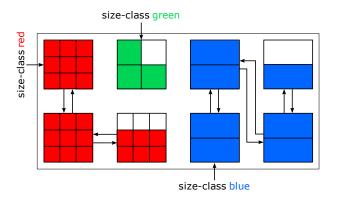
Compact-Fit 000 Partial Compaction

Experiments

Conclusion o

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



Fragmentation Problem

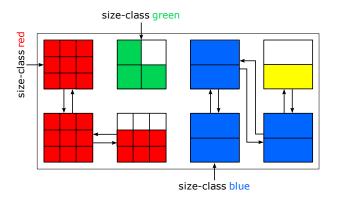
Compact-Fit 000 Partial Compaction

Experiments

Conclusion 0

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



Fragmentation Problem

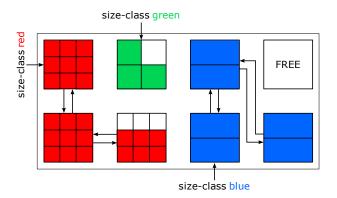
Compact-Fit 000 Partial Compaction

Experiments

Conclusion O

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



Fragmentation Problem

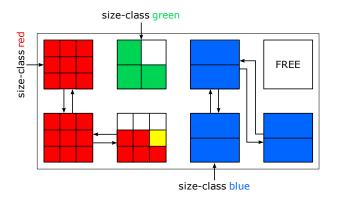
Compact-Fit 000 Partial Compaction

Experiments

Conclusion 0

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



Fragmentation Problem

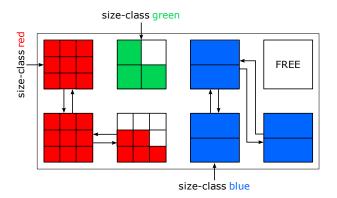
Compact-Fit 000 Partial Compaction

Experiments

Conclusion 0

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



Fragmentation Problem

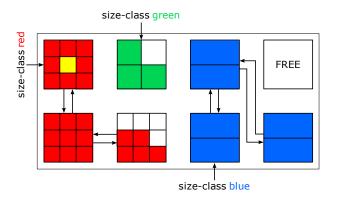
Compact-Fit 000 Partial Compaction

Experiments

Conclusion o

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



Fragmentation Problem

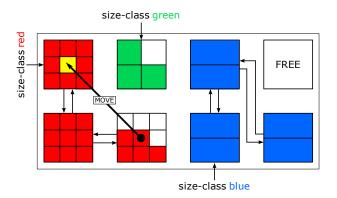
Compact-Fit 000 Partial Compaction

Experiments

Conclusion 0

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



Fragmentation Problem

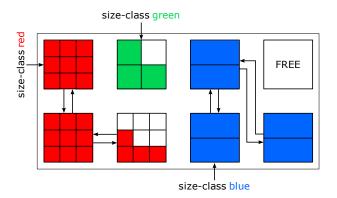
Compact-Fit 000 Partial Compaction

Experiments

Conclusion o

#### Deallocation May Involve Compaction

Size-Class Compact Invariant:



iction	Fragmentation	Pro
	000	
	0000000000	00

n Problem Compact-Fit

Partial Compaction

Experiments

Conclusion O

# Compact-Fit Versions

- Compact-fit moving version (CFM)
  - concrete space = physical memory
  - allocated objects are contiguous in physical memory
  - · compaction: leads to movements in physical memory
- Compact-fit non-moving version (CFNM)
  - concrete space = virtual memory (blocks)
  - allocated objects are not contiguous in physical memory, but are contiguous in virtual memory
  - compaction: reprogramming block table

Compact-Fit

Partial Compaction

Experiments

Conclusion O

# Compact-Fit Moving Version Complexity

- malloc(n) takes Θ(1) time
- free(n) takes O(n) time because of compaction
- memory access (dereference) takes Θ(1) time because of abstract address space
- memory fragmentation is bounded and predictable

◆ロト (母) (量) (量) (量) (量) ( Computational Systems Group, University of Salzburg

Compact-Fit

Partial Compaction

Experiments

Conclusion 0

# Compact-Fit Non-Moving Version Complexity

- malloc(n) takes Θ(n) time because of maintaining the virtual memory
- free(n) takes Θ(n) time because of maintaining the virtual memory and compaction
- memory access (dereference) takes Θ(1) time because of abstract address space and virtual memory
- memory fragmentation is bounded and predictable

Introduction	
00	
00	

Fragmentation Problem

Compact-Fit 000 Partial Compaction •0000000 Experiments 000 Conclusion

# Partial Compaction

#### Idea:

Allow an arbitrary number k of not-full pages within a size-class.

#### Result:

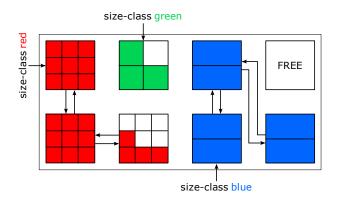
Each deallocation that happens when  $number\_not\_full\_pages \le k$  takes constant time, but fragmentation increases with k.

#### Effect:

This way we formalize, control, and implement the trade-off between temporal performance and memory fragmentation.

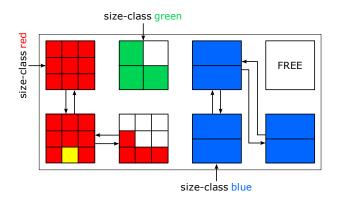
# Partial Compaction

size-class red: k=2



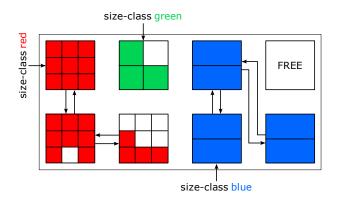
### Partial Compaction

size-class red: k=2



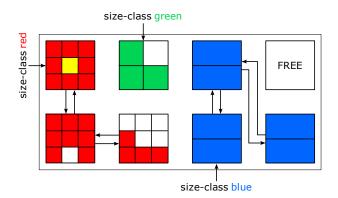
### Partial Compaction

size-class red: k=2



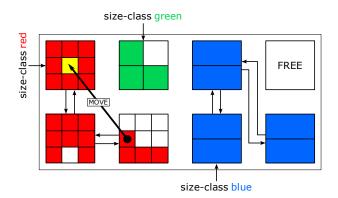
### Partial Compaction

size-class red: k=2



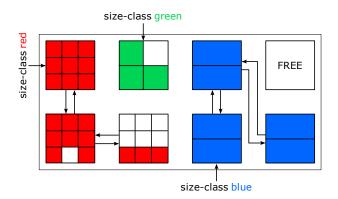
#### Partial Compaction

size-class red: k=2



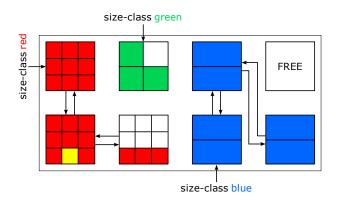
### Partial Compaction

size-class red: k=2



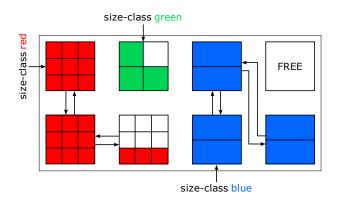
#### Partial Compaction

size-class red: k=2



### Partial Compaction

size-class red: k=2



Introd	uction	
00		
00		

Partial Compaction

Experiments

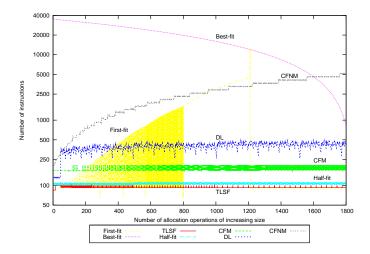
Conclusion O

# Related Work

Tir	ne		
O(1)	First-fit: free, deref Best-fit: free, deref DL: free, deref TLSF: malloc, free, deref Half-fit: malloc, free, deref	CFM: malloc, deref CFNM: deref M e t	
O(log n)		r <sup>0</sup> Jamaica: deref n	
O(n)		o m CFM: free e CFNM: malloc, free Jamaica: malloc, free, deref	
unbounded in n	First-fit: malloc Best-fit: malloc DL: malloc		-Snace
	unpredictable	predictable	-Space

#### ペロト 《□ 》 《■ 》 《 ■ 》 《 ■ 》 ■ ○ へ() Computational Systems Group, University of Salzburg

#### Incremental Allocation Benchmark



Computational Systems Group, University of Salzburg

・ロト ・ 日 ・ ・ ヨ ・ ・ ヨ ・

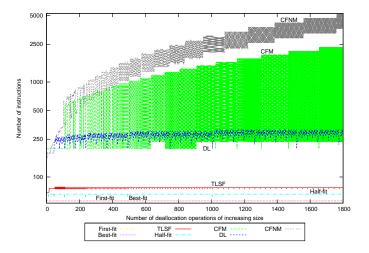
ct-Fit P

Partial Compaction

Experiments

Conclusion O

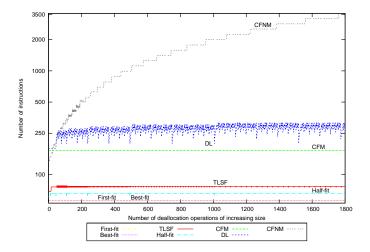
#### Incremental Free Benchmark



Computational Systems Group, University of Salzburg

・ロト ・回 ト ・ヨト ・ヨト

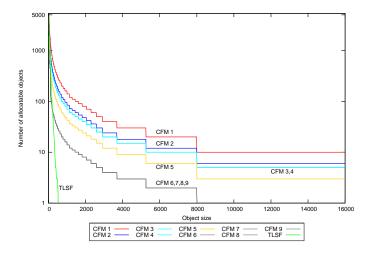
#### Incremental Free Partial Compaction



・ロト・日本・モン・モン・モン・モン・マーン
Computational Systems Group, University of Salzburg

Introduction	Fragmentation Problem	Compact-Fit	Partial Compaction	Experiments	Conclusion
00 00	000 00000000000000000000000000000000000	000	00000000	000	0

#### Fragmentation



・ロン ・四 と ・ ヨ と ・ モ と Computational Systems Group, University of Salzburg

Э

Partial Compaction

Experiments

Conclusion

# Conclusion and Future Work

Contribution:

- Compact-fit is predictable in time and space
- moving and non-moving Compact-fit implementations

Future work:

- virtual machine implementation
- source-to-source translator
- concurrency and multi-processor support
- static program analysis can help to optimize the k for the partial compaction strategy

```
http://tiptoe.cs.uni-salzburg.at/compact-fit
```