A Lock-Free Dynamically Resizable Array

Damian Dechev[1] Peter Pirkelbauer[1] Bjarne Stroustrup[1,2]

[1]Department of Computer Science
Texas A&M University

[2]AT&T Research

OPODIS, December 2006

Parasol
Smarter computing.

Overview

A first design of a lock-free [Fra04] dynamically resizable array for C++

- Similar to STL vector [Str00].
- Dynamic memory management
- Constant-time random access

Parasol
Smarter computing.

Motivation

- STL containers used with locks
  - Performance problems (low parallelism)
  - Safety problems (deadlock, livelock, priority inversion)
- Application Areas
  - Autonomous Real-Time Systems
  - Mission Data Systems at JPL
  - Distributed Parallel Containers [AJR+01]

Parasol
Smarter computing.

Design Principles

- Portability
  - word-size compare and swap (CAS)
- Efficiency
  - Minimal Overhead
  - Wait-free [Fra04] random access read and write
- Linearizable operations [HW90]
- Lock-free memory allocation and management

Parasol
Smarter computing.

std::vector goes lockfree

- Complexity of operation:

Operation	modified Location
resize	all entries
push_back	size, tail
pop_back	size
write	one element
read	none
size	none

- resize: relocates all elements and alters the capacity
  - Two-level array
- push_back: modifies size and an element
  - Barnes-style announcement [Bar93]
- element-size: pointer semantics

Parasol
Smarter computing.

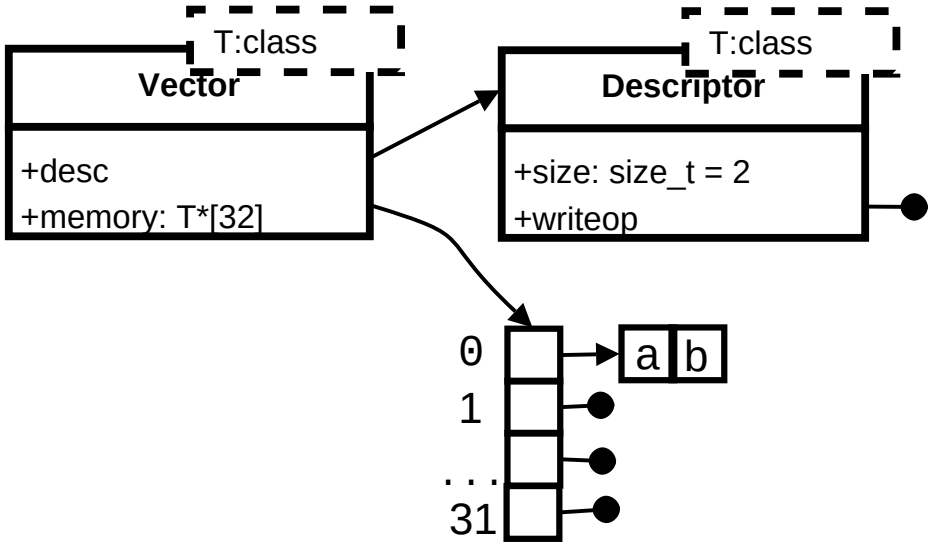
lockfree::vector - interface

Semantics

- Assumes sequential semantics
  - vec.back(); vec.pop_back();
- pop_back: removes the last element
  - returns *and* removes the last element

Parasol
Smarter computing.

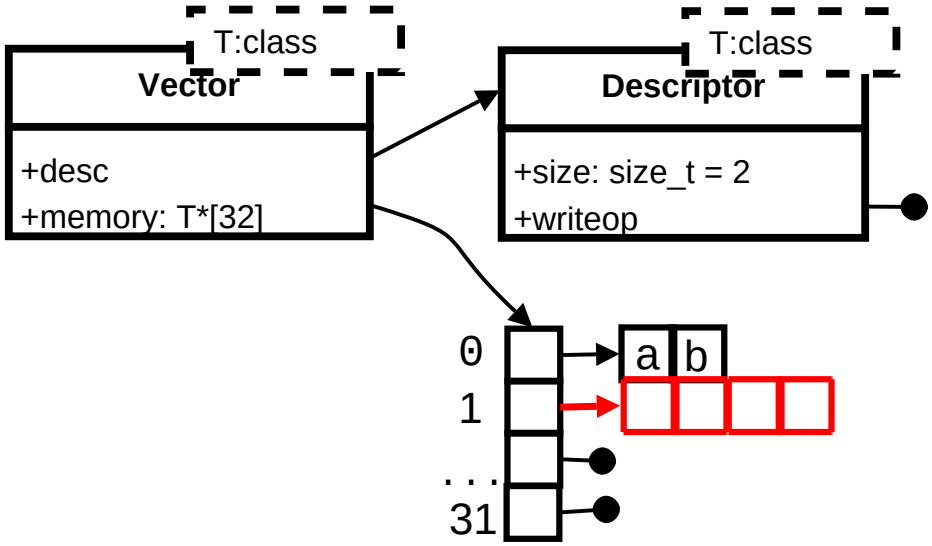
lockfree::vector - push_back




```
push_back(vec, e)

do {
  Desc desc = vec.desc

  complete_write(vec, desc)
  ensure_capacity(vec, desc.size+1)

  WriteDesc wd = createWriteDesc(vec,desc,e)
  Desc newdesc = createDesc(vec, desc, wd)
} while !CAS(&vec.desc, desc, newdesc)

complete_write(newdesc)
```

Parasol
Smarter computing.

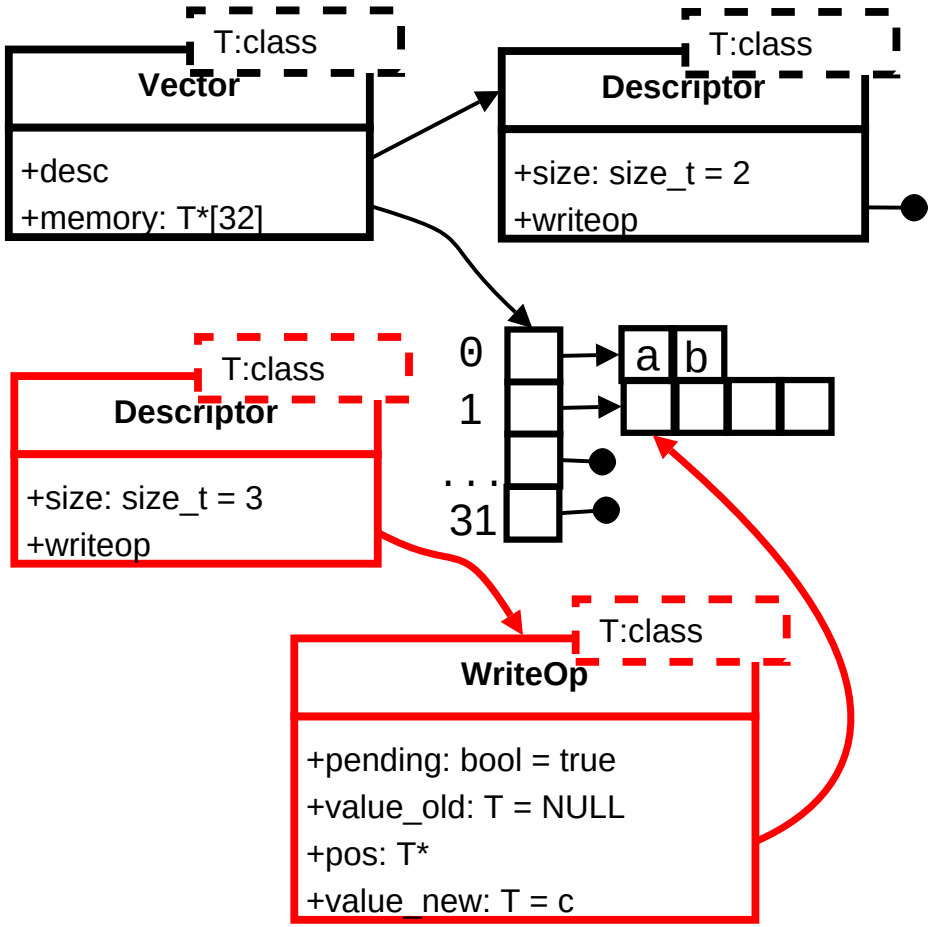
lockfree::vector - push_back (cont'd)




```
push_back(vec, e)

do {
  Desc desc = vec.desc

  complete_write(vec, desc)
  ensure_capacity(vec, desc.size+1)

  WriteDesc wd = createWriteDesc(vec,desc,e)
  Desc newdesc = createDesc(vec, desc, wd)
} while !CAS(&vec.desc, desc, newdesc)

complete_write(newdesc)
```

Parasol
Smarter computing.

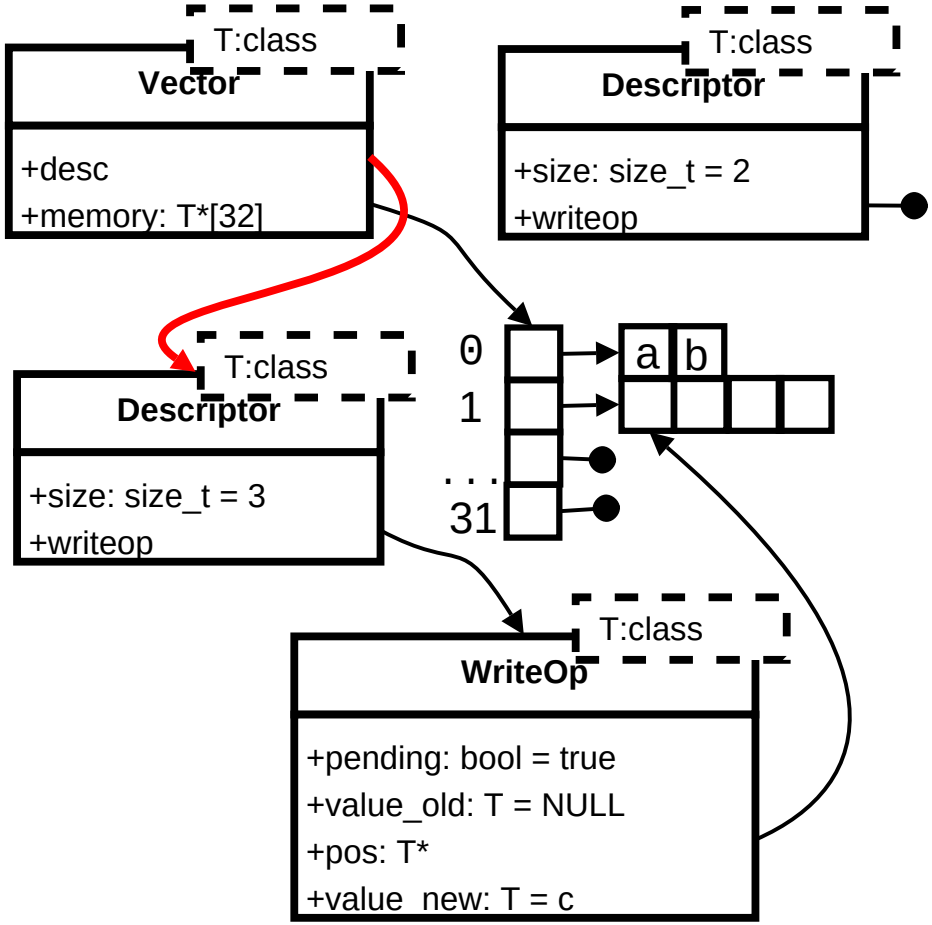
lockfree::vector - push_back (cont'd)




```
push_back(vec, e)

do {
  Desc desc = vec.desc

  complete_write(vec, desc)
  ensure_capacity(vec, desc.size+1)

  WriteDesc wd = createWriteDesc(vec,desc,e)
  Desc newdesc = createDesc(vec, desc, wd)
} while !CAS(&vec.desc, desc, newdesc)

complete_write(newdesc)
```

Parasol
Smarter computing.

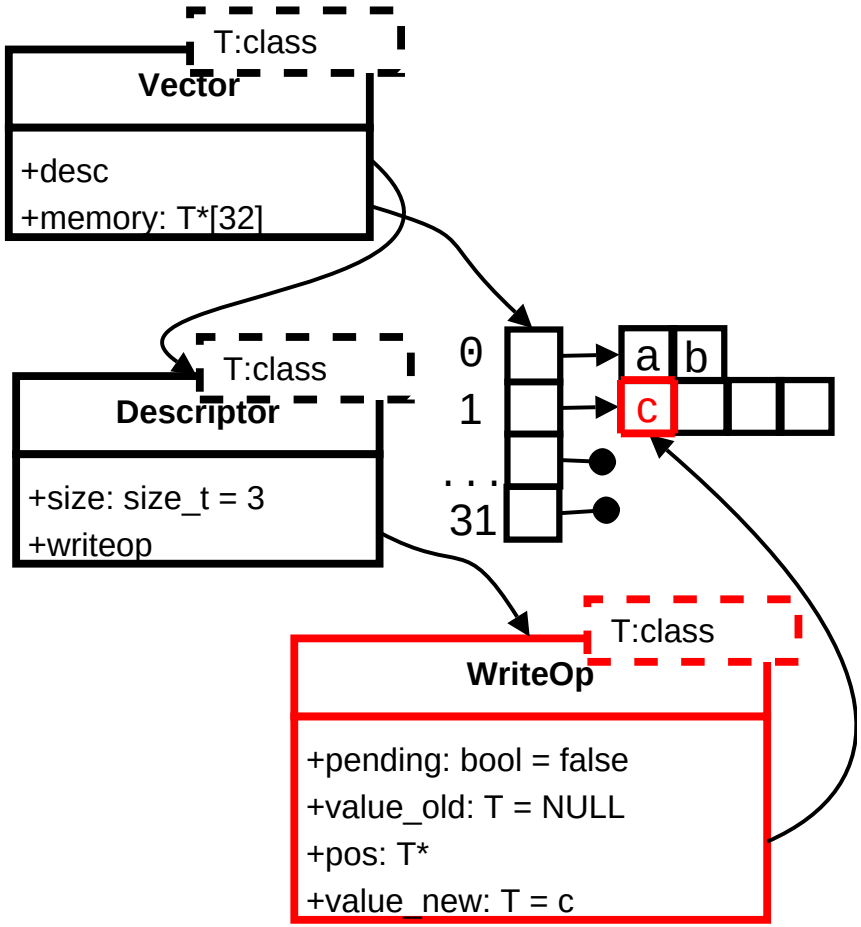
lockfree::vector - push_back (cont'd)




```
push_back(vec, e)

do {
  Desc desc = vec.desc

  complete_write(vec, desc)
  ensure_capacity(vec, desc.size+1)

  WriteDesc wd = createWriteDesc(vec,desc,e)
  Desc newdesc = createDesc(vec, desc, wd)
} while !CAS(&vec.desc, desc, newdesc)

complete_write(newdesc)
```

Parasol
Smarter computing.

lockfree::vector - push_back (cont'd)




```
push_back(vec, e)

do {
  Desc desc = vec.desc

  complete_write(vec, desc)
  ensure_capacity(vec, desc.size+1)

  WriteDesc wd = createWriteDesc(vec,desc,e)
  Desc newdesc = createDesc(vec, desc, wd)
} while !CAS(&vec.desc, desc, newdesc)

complete_write(newdesc)
```

Parasol
Smarter computing.

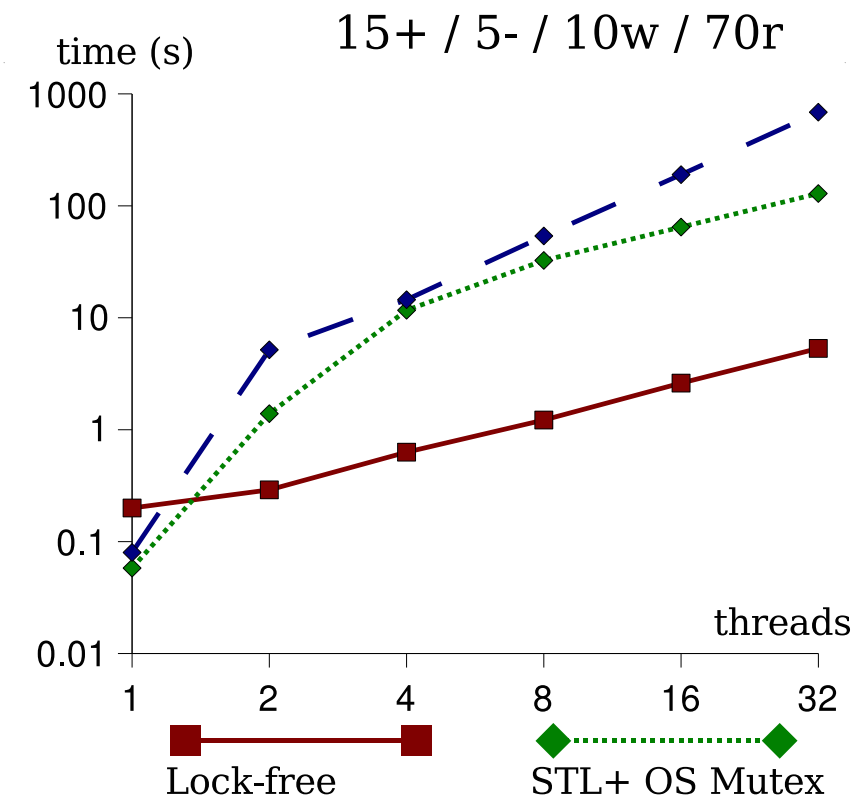
lockfree::vector - Operations

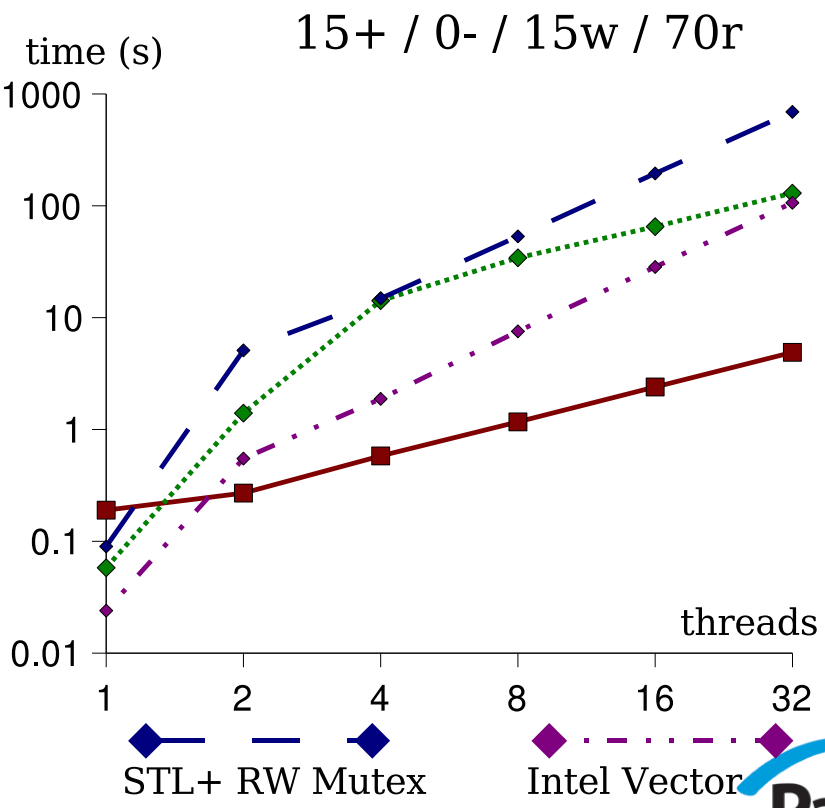
	Operations	Descriptor (Desc)	Complexity
push_back	$Vec \times Elem \longrightarrow void$	$Desc_t \longrightarrow Desc_{t+1}$	$O(1) \times cong.$
pop_back	$Vec \longrightarrow Elem$	$Desc_t \longrightarrow Desc_{t+1}$	$O(1) \times cong.$
resize	$Vec \times size\_t \longrightarrow Vec$	$Desc_t \longrightarrow Desc_t$	$O(1)$
read	$Vec \times size\_t \longrightarrow Elem$	$Desc_t \longrightarrow Desc_t$	$O(1)$
write	$Vec \times size\_t \times Elem \longrightarrow Vec$	$Desc_t \longrightarrow Desc_t$	$O(1)$
size	$Vec \longrightarrow size\_t$	$Desc_t \longrightarrow Desc_t$	$O(1)$

Parasol
Smarter computing.

lockfree::vector - Performance (Dual-Core)

- Intel 1.83GHz Dual Core
  - 512MB shared RAM, 2MB shared L2 cache, MAC OS X

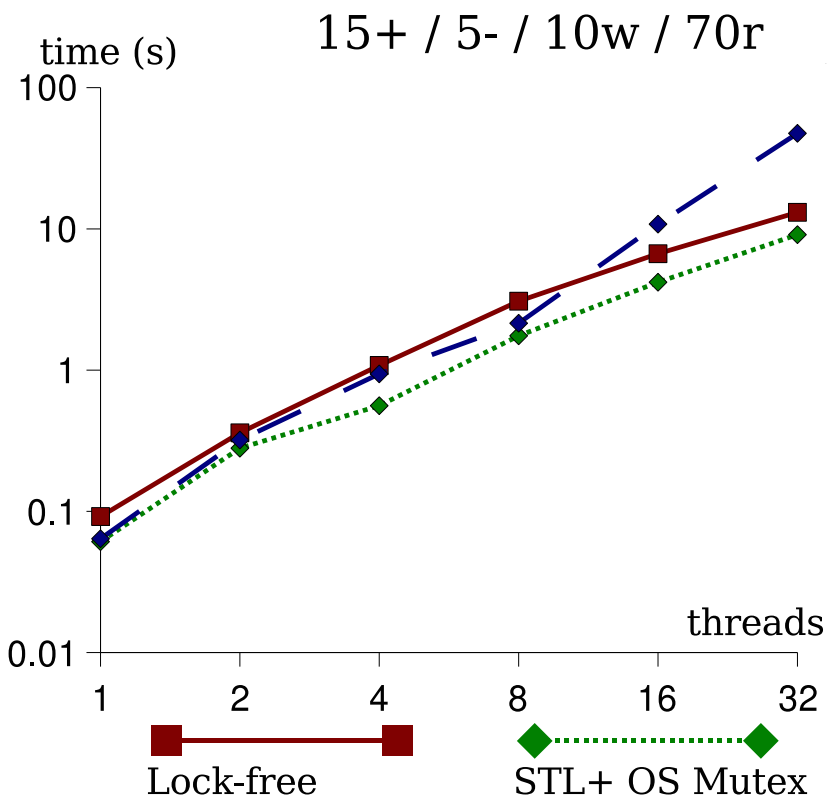


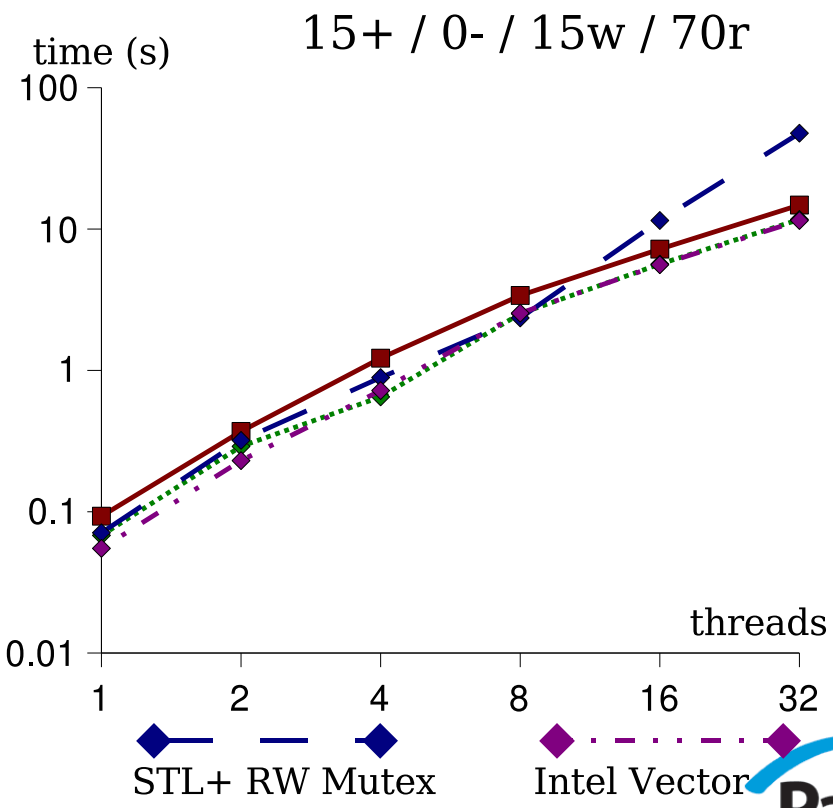


Parasol
Smarter computing.

lockfree::vector - Performance (4xDual-Core)

- AMD 2.2GHZ Quad Dual Core Opteron
  - 4GB shared RAM, 1MB L2 cache (per die), MS Windows 2003

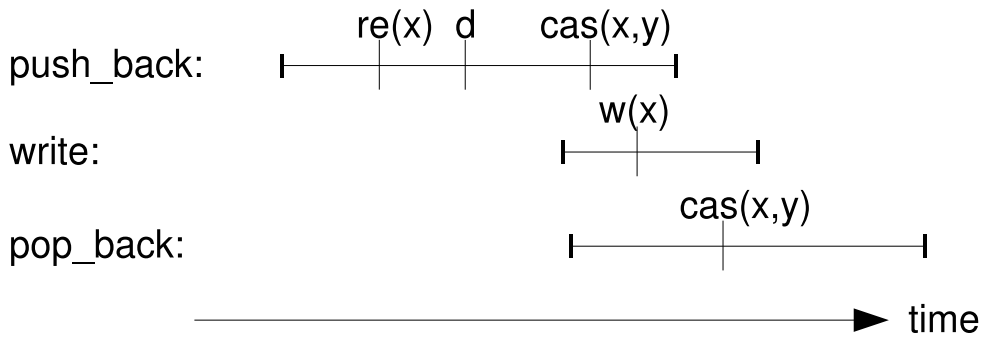




Parasol
Smarter computing.

lockfree::vector - ABA Problem

- Common to all CAS-based systems
- General Solution:
  - version counter



Parasol
Smarter computing.

lockfree::vector - ABA Problem (cont'd)

- Prevention: store unqiue elements
- Solution:
  - Object Reclamation Scheme
    - ★ Pass The Buck (Herlihy et al. [HLMM05])
    - ★ Hazard Pointers (Michael [Mic04])
  - Implement Value Semantics

Parasol
Smarter computing.

Conclusion and Future Work

- First practical and portable design of a lock-free vector
- Future Work
  - Integrate effective ABA solution
  - Refine lockfree::vector interfaces

Parasol
Smarter computing.

References I

Ping An, Alin Jula, Silvius Rus, Steven Saunders, Tim Smith, Gabriel Tanase, Nathan Thomas, Nancy Amato, and Lawrence Rauchwerger.
STAPL: A Standard Template Adaptive Parallel C++ Library.
In *LCPC '01*, pages 193–208, Cumberland Falls, Kentucky, Aug 2001.

Greg Barnes.
A method for implementing lock-free shared-data structures.
In *SPAA '93: Proceedings of the fifth annual ACM symposium on Parallel algorithms and architectures*, pages 261–270, New York, NY, USA, 1993. ACM Press.

Keir Fraser.
Practical lock-freedom.
Technical Report UCAM-CL-TR-579, University of Cambridge, Computer Laboratory, February 2004.

Maurice Herlihy, Victor Luchangco, Paul Martin, and Mark Moir.
Nonblocking memory management support for dynamic-sized data structures.
ACM Trans. Comput. Syst., 23(2):146–196, 2005.

Maurice P. Herlihy and Jeannette M. Wing.
Linearizability: a correctness condition for concurrent objects.
ACM Trans. Program. Lang. Syst., 12(3):463–492, 1990.

Maged M. Michael.
Hazard Pointers: Safe Memory Reclamation for Lock-Free Objects.
IEEE Trans. Parallel Distrib. Syst., 15(6):491–504, 2004.

Bjarne Stroustrup.
The C++ Programming Language.
Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

Parasol
Smarter computing.