

Optimization of allocators for POD- types

By John Juul Jensen

5th STL Workshop, 21 June 2005

What's it all about

- STL vectors offer amortized constant time insertion, by doubling the capacity whenever it's full
- Vectors also offer contiguous memory, which means that it obeys the identity `&v[n] == &v[0] + n` for all `0 <= n < v.size()`
- When the vector needs more memory it allocates a new block and copies the existing data

Ideas to speed up `std::vector`

- Use `realloc` to expand existing block of memory instead of allocating a new block of memory
 - Pro: Reduces the need for copying
 - Pro: `realloc` handles copying whenever it's needed
 - Con: Not all classes are bitwise moveable

What classes are bitwise moveable

- POD (Plain old datatype) (has no constructor)
- Classes to which there are no references
 - References into vectors are not a good idea since insert/deletion might invalidate references
 - Classes to which there are references, must maintain these references during (copy)construction to function properly within an STL container

How can this be exploited

- It can't
- STL handles all copying of objects by copyconstruction within the containers, effectively disallowing users to optimize this process
- Allocators offers no solution, since they only deal with memory allocation

Rolling your own vector

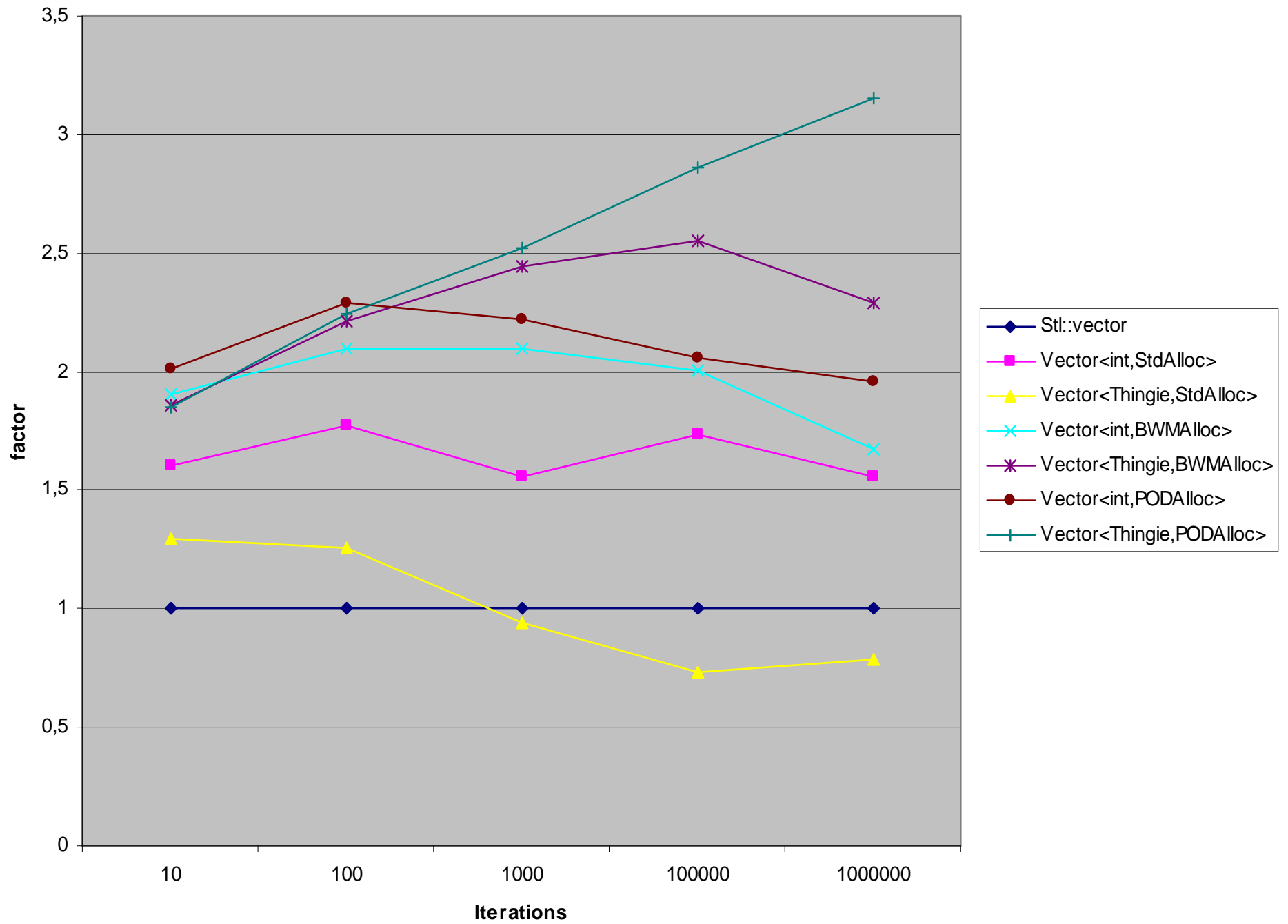
- Creating a new vector that supports both BW-Moveable types and types that need copyconstruction solves the problem, but is not STL compatible
- In the new vector the allocator takes care of memory allocation and copies objects
- Creating different allocators, allows us to optimize vector for a given type

Ideas for implementation

- Type traits
 - IsPod<T>
 - IsBWMoveable<T>
- Enables us to automatically select an allocator at compile-time
- Three different kinds of allocators
 - StdAllocator
 - BWMoveableAllocator
 - PODAllocator

Benchmarking the new vector

Loops	Type	Stl::vector	Vector<StdAlloc>	Vector<BWMAAlloc>	Vector<PODAAlloc>
10	Int	1	1,60563	1,90795	2,00881
10	Thingie	1	1,29206	1,85845	1,85
100	Int	1	1,77103	2,09972	2,29003
100	Thingie	1	1,25734	2,21277	2,24
1000	Int	1	1,55667	2,09471	2,22324
1000	Thingie	1	0,940233	2,4401	2,52282
100000	Int	1	1,73116	2,00134	2,06169
100000	Thingie	1	0,732869	2,54844	2,85905
1000000	Int	1	1,56026	1,66913	1,96163
1000000	Thingie	1	0,784999	2,2927	3,15044



REPS = 10 LOOPS = 10

A: std::vector<int>>::push_back 0,00012739 Seconds

B: Vector<int,StdAllocator<int> >::push_back 7,93397e-005 Seconds

C: Vector<int,BWMovableAllocator<int> >::push_back 6,67683e-005 Seconds

D: Vector<int,PODAllocator<int> >::push_back 6,67683e-005 Seconds

A/B = 1,60563

A/C = 1,90795

A/D = 2,00881

E: std::vector<Thingie>::push_back 0,000113702 Seconds

F: Vector<Thingie,StdAllocator<Thingie> >::push_back 8,8e-005 Seconds

G: Vector<Thingie,BWMovableAllocator<Thingie> >::push_back 6,1181e-005 Seconds

H: Vector<Thingie,PODAllocator<Thingie> >::push_back 6,14603e-005 Seconds

E/F = 1,29206

E/G = 1,85845

E/H = 1,85

REPS = 10 LOOPS = 100

A: std::vector<int>>::push_back 0,000211759 Seconds

B: Vector<int,StdAllocator<int> >::push_back 0,000119568 Seconds

C: Vector<int,BWMovableAllocator<int> >::push_back 0,000100851 Seconds

D: Vector<int,PODAllocator<int> >::push_back 0,000100851 Seconds

A/B = 1,77103

A/C = 2,09972

A/D = 2,29003

E: std::vector<Thingie>::push_back 0,000203378 Seconds

F: Vector<Thingie,StdAllocator<Thingie> >::push_back 0,000161752 Seconds

G: Vector<Thingie,BWMovableAllocator<Thingie> >::push_back 9,19111e-005 Seconds

H: Vector<Thingie,PODAllocator<Thingie> >::push_back 9,07937e-005 Seconds

E/F = 1,25734

E/G = 2,21277

E/H = 2,24

REPS = 10 LOOPS = 1000

A: std::vector<int>>::push_back 0,000475759 Seconds

B: Vector<int,StdAllocator<int> >::push_back 0,000305625 Seconds

C: Vector<int,BWMovableAllocator<int> >::push_back 0,000227124 Seconds

D: Vector<int,PODAllocator<int> >::push_back 0,000227124 Seconds

A/B = 1,55667

A/C = 2,09471

A/D = 2,22324

E: std::vector<Thingie>::push_back 0,000540571 Seconds

F: Vector<Thingie,StdAllocator<Thingie> >::push_back 0,000574933 Seconds

G: Vector<Thingie,BWMovableAllocator<Thingie> >::push_back 0,000221537 Seconds

H: Vector<Thingie,PODAllocator<Thingie> >::push_back 0,000214273 Seconds

E/F = 0,940233

E/G = 2,4401

E/H = 2,52282

REPS = 10 LOOPS = 100000

A: std::vector<int>>::push_back 0,0308872 Seconds

B: Vector<int,StdAllocator<int> >::push_back 0,0178419 Seconds

C: Vector<int,BWMovableAllocator<int> >::push_back 0,0154332 Seconds

D: Vector<int,PODAllocator<int> >::push_back 0,0154332 Seconds

A/B = 1,73116

A/C = 2,00134

A/D = 2,06169

E: std::vector<Thingie>::push_back 0,0448448 Seconds

F: Vector<Thingie,StdAllocator<Thingie> >::push_back 0,0611907 Seconds

G: Vector<Thingie,BWMovableAllocator<Thingie> >::push_back 0,0175969 Seconds

H: Vector<Thingie,PODAllocator<Thingie> >::push_back 0,0156852 Seconds

E/F = 0,732869

E/G = 2,54844

E/H = 2,85905

REPS = 10 LOOPS = 1000000

A: std::vector<int>>::push_back 0,205088 Seconds

B: Vector<int,StdAllocator<int> >::push_back 0,131445 Seconds

C: Vector<int,BWMovableAllocator<int> >::push_back 0,122871 Seconds

D: Vector<int,PODAllocator<int> >::push_back 0,122871 Seconds

A/B = 1,56026

A/C = 1,66913

A/D = 1,96163

E: std::vector<Thingie>::push_back 0,344952 Seconds

F: Vector<Thingie,StdAllocator<Thingie> >::push_back 0,439429 Seconds

G: Vector<Thingie,BWMovableAllocator<Thingie> >::push_back 0,150457 Seconds

H: Vector<Thingie,PODAllocator<Thingie> >::push_back 0,109493 Seconds

E/F = 0,784999

E/G = 2,2927

E/H = 3,15044