

Looking in boost code:

```
class X{  
public:
```

```
    explicit X(int n): n_(n){  
    }  
  
    void * operator new(std::size_t){  
        return std::allocator<X>().allocate(1, static_cast<X*>(0));  
    }  
  
    void operator delete(void * p){  
        std::allocator<X>().deallocate(static_cast<X*>(p), 1);  
    }  
}
```

```
private:
```

```
    X(X const &);  
    X & operator=(X const &);  
  
    int n_;  
};
```

or

```
class Y{  
public:
```

```
    explicit Y(int n): n_(n){  
    }  
  
    void * operator new(std::size_t n){  
        return boost::detail::quick_allocator<Y>::alloc(n);  
    }  
  
    void operator delete(void * p, std::size_t n){  
        boost::detail::quick_allocator<Y>::dealloc(p, n);  
    }  
}
```

```
private:
```

```
    Y(Y const &);  
    Y & operator=(Y const &);  
  
    int n_;
```

```
};
```

where something in the memory allocator we can get alignmet of type by:

for Codegear: `alignof(T)`

for GCC: `__alignof__(T)`

for MSC: `__alignof(T)`

according by IBM link and boost source code.

According by boost source code in file `intrinsics.hpp` MSC `__alignof(T)` fails when used with `/Zp` property. We need to be care of.

In this case we can easily use `sse2/3/4/...`:

```
Y y_v = new Y(1);
```

Is possible to implement a tool that can be integrated in the "operator new" of the classes with `sse2/3/4` types properties?

Sorry, if it is a stupid question. I have not experience with sse, but i'me very interested to speed up the program by using `sse/2/3/4`.