Hidden Copies

- Even moves can result in hidden costs

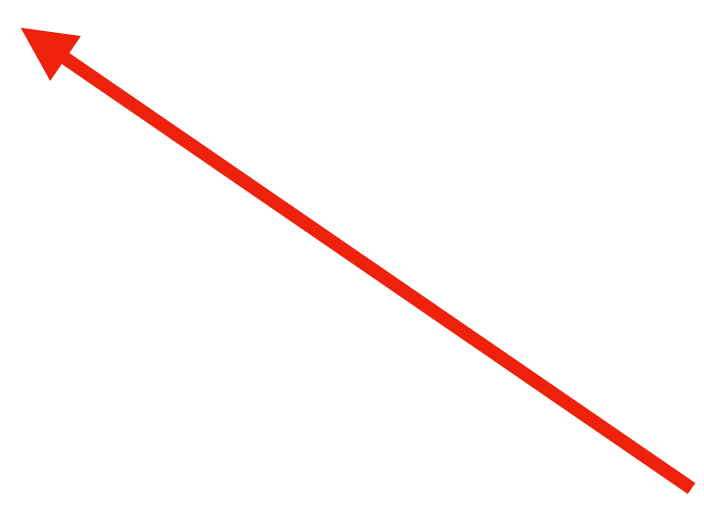
- Especially when dealing with legacy code which may not have proper move constructors/assignment operators

```
class LegacyObject
{
public:
    LegacyObject();
    LegacyObject (const LegacyObject&);
    LegacyObject& operator=(const LegacyObject&);
    ...
};

struct Parent
{
    std::vector<int> vec;
    LegacyObject obj;
};
```

```
int main()
{
    Parent a, b;
    a = std::move (b);

    return 0;
}
```



`LegacyObject` has no move operator so the copy assignment operator will be called

12

Hidden Copies

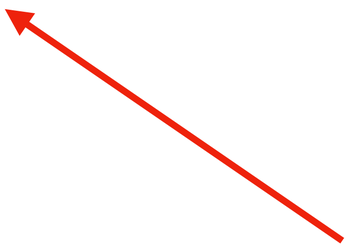
- Even moves can result in hidden costs
- Especially when dealing with legacy code which may not have proper move constructors/assignment operators

```
class LegacyObject
{
public:
    LegacyObject();
    LegacyObject (const LegacyObject&);
    LegacyObject& operator=(const LegacyObject&);
    ...
};

struct Parent
{
    std::vector<int> vec;
    LegacyObject obj;
};
```

```
int main()
{
    Parent a, b;
    a = std::move (b);

    return 0;
}
```



`LegacyObject` has no move operator so the copy assignment operator will be called

Blocking vs. Non-wait-free vs. Wait Free

Blocking	Non-wait-free	Wait-free
May context switch for example due to a lock, system call etc.	Execution time is unbounded	Execution time is bounded*
Caches likely to be invalidated	Must contain a loop (which is unbounded)	No unbounded loops
Memory may be swapped	Blocking operations are never wait-free (but not vice versa)	