代码片段

```
using namespace concurrency;
struct C{
void operator() (){ ...}
};
C func;
parallel_invok(func);
```

编译错误信息

```
…\include\ppl.h(127): error C3848: expression having type `const C'
would lose const-volatile qualifiers in order to call `void C::operator ()()'
…\include\concrt.h(4378) : see reference to function template instantiation
`void Concurrency::task_handle<_Function1>::operator ()(void) const' being compiled
…\include\ppl.h(948) : see reference to function template instantiation
`void Concurrency::_Parallel_invoke_impl<_Function1,_Function2>
(const _Function1 &,const _Function2 &)' being compiled
```

用户意见

If I make the C::operator()() const, I lose a lot of the benefits of a function object, namely, that its state is maintained internally between calls. Is there a way that I could invoke non-const function objects in parallel?