TThreadedObject

The Bug

```
const auto imtPoolSize = ROOT::GetImplicitMTPoolSize();
fMaxSlots = (64 > imtPoolSize) ? fgMaxSlots : imtPoolSize;
```

Note: fgMaxSlots == 64

The ~~Bug~~ Feature?!

```
const auto imtPoolSize = ROOT::GetImplicitMTPoolSize();
fMaxSlots = (64 > imtPoolSize) ? fgMaxSlots : imtPoolSize;
```

No IMT means 64 slots! Several tests have IMT off - it works!

"...or at least 1": some tests use [0],[1] and it works!

Some tests start some threads without IMT - and it works!

What do we need?

- What if I want 65 threads, but no IMT?
  - Only I, constructing TThreadedObject, know.
- nSlots cannot be changed later. And that's probably a good thing.
- Thus, "nSlots" should be a constructor argument

Getting "nSlots" into TTO (1)

- TThreadedObject takes a parameter pack: TThreadedObject(ARGS...)
- Can have

```
template <class... Args> TTO(Args... args);
template <class... Args> TTO(int nSlots, Args... args);
```

  - But is TTO(10, 0., 1.)
    - 10 slots, passing (0.,1.) to its T, or
    - Relying on IMT and passing (10, 0., 1.) to its T?

Getting "nSlots" into TTO (2)

- Can have

```
template <class... Args> TTO(Args... args);
template <class... Args> TTO(TTOSlots_t, Args... args);
```

  - Then TTO(10, 0., 1) calls T(10, 0., 1),
    TTO(TTOSlots_t(10), 0., 1.) calls T(0., 1.)

Getting "nSlots" into TTO (3)

- Or do we expect the number of slots to be known at compile time?

- Can have

```
template <class T, int NSlots> class TTO {
   template <class... Args> TTO(Args... args);
```

- But do we really know nSlots at compile time? (Think "IMT cores")

Suppose we find a way

- Say we settle for Can have
```
template <class... Args> TTO(Args... args);
template <class... Args> TTO(Slots, Args... args);
```
- Then what do we do here:
```
EnableIMT(4); ...; TTO o(Slots(N)); /*o with IMT?*/
```
- And here:
```
EnableIMT(); ...; TTO o(Slots(N)); /*o with IMT?*/
```
- I.e. is "IMT slots" a special nSlots ("-1"), and we should complain / throw if a different nSlots is provided?

Capping NSlots

- Currently, nSlots has a minimum of 64
- I assume it's a bug and was meant to be *maximum* of 64
- But what about
  `EnableIMT(/*256*/); /*use TThreadedExecutor-with-TTO*/`
- I.e. can we cap, at all?