Study creational design patterns from production code

SWPP TEAM1

**ABSTRACT FACTORY PATTERN**

: SparseConditionalAccumulator extends TypedConditionalAccumulatorBase which

extends ConditionalAccumulatorBase

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/kernels/conditional\_accumulator\_base.h

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/kernels/typed\_conditional\_accumulator\_base.h

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/kernels/sparse\_conditional\_accumulator.h

ConditionalAccumulatorBase has virtual function

details in the comment in the code

protected:

// Virtual methods to be implemented by sub-classes for different datatypes.

// Implements arithmetic operations specific to datatype.

virtual void DivideAccumGradByCounter(OpKernelContext\* ctx)

EXCLUSIVE\_LOCKS\_REQUIRED(mu\_) = 0;

virtual bool SetOutput(OpKernelContext\* ctx) = 0;

**BUILDER PATTERN**

NodeBuilder

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/graph/node\_builder.cc

NodeBuilder

NodeBuilder& NodeBuilder::ControlInput(Node\* src\_node) {

control\_inputs\_.emplace\_back(src\_node);

def\_builder\_.ControlInput(src\_node->name());

return \*this;

}

NodeBuilder& NodeBuilder::ControlInputs(gtl::ArraySlice<Node\*> src\_nodes) {

control\_inputs\_.insert(control\_inputs\_.end(), src\_nodes.begin(),

src\_nodes.end());

for (const Node\* src\_node : src\_nodes) {

def\_builder\_.ControlInput(src\_node->name());

}

return \*this;

}

**DEPENDENCY INJECTION**

<https://github.com/apache/reef/blob/master/lang/cs/Org.Apache.REEF.Common/Runtime/Evaluator/Context/RootContextLauncher.cs>

[Inject]

private RootContextLauncher(

AvroConfigurationSerializer serializer,

[Parameter(typeof(RootContextConfiguration))] string rootContextConfiguration,

[Parameter(typeof(RootServiceConfiguration))] string rootServiceConfiguration,

[Parameter(typeof(InitialTaskConfiguration))] string initialTaskConfiguration,

IInjector injector)

: this(serializer.FromString(rootContextConfiguration),

serializer.FromString(rootServiceConfiguration),

Optional<IConfiguration>.Of(serializer.FromString(initialTaskConfiguration)),

injector)

{

}