Femtocode: querying HEP data

Jim Pivarski

Princeton University – DIANA

April 17, 2017

Features of the event loop

- JIT-compiled for the specific nesting of loops.
-

Using ROOT functions in Femtocode

```
# ROOT/some_library.py, somewhere visible to Femtocode client and server.
import ctypes
libMathCore = ctypes.cdll.LoadLibrary("libMathCore.so")
chi2_ctypes = libMathCore._ZN5TMath17ChisquareQuantileEdd    # c++filt!
chi2_ctypes.argtypes = (ctypes.c_double, ctypes.c_double)
chi2_ctypes.restype = ctypes.c_double
```

```
###################################################### Creating a custom library
from femtocode.typesystem import *
from femtocode.lib.custom import *
from femtocode.run.execution import NativeTestSession

def chi2_sig(x, n):
    # the function's type signature; provides compile-time type-safety
    assert isinstance(x, Number) and \
        almost.min(0, x.min) == 0 and almost.max(x.max, almost(1)) == almost(1)
    assert isinstance(n, Number) and n.whole and n.min > 0
    return real(0, 1)

custom = CustomLibrary()
custom.add(CustomFlatFunction("chi2", "ROOT.some_library", "chi2_ctypes", chi2_sig))

##################################################################### Testing it
# Define a dataset with the right types and fill it with dummy data.
session = NativeTestSession()
numerical = session.source("Test", x=real(0, almost(1)), n=integer(1, almost(inf)))
for i in xrange(100):
    numerical.dataset.fill({"x": i / 100.0, "n": i + 1})

# Femtocode calls TMath::ChisquareQuantile without involving Python at all.
result = numerical.toPython(out = "chi2(x, n)").submit(libs=[custom])
for entry in result:
    print entry
```