Unified diff: `step4_pi` *vs.* `step5_pi_checkpoint`

```
Files step4_pi/CMakeLists.txt and step5_pi_checkpoint/CMakeLists.txt are identical

diff -N -u -r -b -s step4_pi/main.cpp step5_pi_checkpoint/main.cpp
--- step4_pi/main.cpp   2016-06-15 14:36:39.478852887 -0400
+++ step5_pi_checkpoint/main.cpp        2016-06-15 14:12:52.874919385 -0400
@@ -22,14 +22,29 @@

     mysim_type mysim(p);

+    // If needed, restore the last checkpoint
+    std::string checkpoint_file = p["checkpoint"].as<std::string>();
+
+    if (p.is_restored()) {
+        std::cout << "Restoring checkpoint from " << checkpoint_file
+                  << std::endl;
+        mysim.load(checkpoint_file);
+    }
+
     std::cout << "Starting simulation"
               << std::endl;

     mysim.run(alps::stop_callback(std::size_t(p["timelimit"])));

     std::cout << "Simulation finished"
-              << std::endl
-              << "Collecting results..."
+              << std::endl;
+
+    std::cout << "Saving to checkpoint " << checkpoint_file
+              << std::endl;
+
+    mysim.save(checkpoint_file);
+
+    std::cout << "Collecting results..."
               << std::endl;

     alps::accumulators::result_set results=mysim.collect_results();
@@ -57,5 +72,12 @@
               << pi_res.autocorrelation<double>()
               << std::endl;

+    // Saving to the output file
+    std::string output_file = p["outputfile"];
+    std::cout << "Saving results to " << output_file << std::endl;
+    alps::hdf5::archive ar(output_file, "w");
+    ar["/parameters"] << p;
+    ar["/simulation/results"] << results;
+
     return 0;
 }
```

```
diff -N -u -r -b -s step4_pi/simulation.cpp step5_pi_checkpoint/simulation.cpp
--- step4_pi/simulation.cpp     2016-06-15 12:31:08.191203939 -0400
+++ step5_pi_checkpoint/simulation.cpp  2016-06-15 12:46:51.367159975 -0400
@@ -1,5 +1,6 @@
 #include "simulation.hpp"
 #include <cmath>
+#include <alps/params/convenience_params.hpp>

 bool MySimulation::is_inside_area(double x, double y) {
     // Let it be just a 1x1 square centered at (0,0)
@@ -58,6 +59,10 @@
 }

 MySimulation::parameters_type& MySimulation::define_parameters(MySimulation::parameters_type& params) {
+    // Do not redefine if we are restoring from the checkpoint:
+    if (params.is_restored()) return params;
+    // Add convenience parameters:
+    alps::define_convenience_parameters(params);
     // Parameters defined by base class:
     return alps::mcbase::define_parameters(params)
         // and by our class:
@@ -66,3 +71,31 @@
         .define<long>("burn-in", 10000, "Number of steps before taking measurements")
         .define<double>("step", "Maximum size of a trial step");
 }
+
+// Saves the state to the hdf5 file
+void MySimulation::save(alps::hdf5::archive & ar) const {
+    // Most of the save logic is already implemented in the base class
+    alps::mcbase::save(ar);
+
+    // We just need to add our own internal state
+    ar["checkpoint/istep"] << istep_;
+    ar["checkpoint/x"] << x_;
+    ar["checkpoint/y"] << y_;
+    // The rest of the internal state is saved as part of the parameters
+}
+
+// Loads the state from the hdf5 file
+void MySimulation::load(alps::hdf5::archive & ar) {
+    // Most of the load logic is already implemented in the base class
+    alps::mcbase::load(ar);
+
+    // Restore the internal state that came from parameters
+    burnin_ = parameters["burn-in"];
+    maxcount_ = parameters["count"];
+    stepsize_ = parameters["step"];
+
+    // Restore the rest of the state from the hdf5 file
+    ar["checkpoint/istep"] >> istep_;
+    ar["checkpoint/x"] >> x_;
+    ar["checkpoint/y"] >> y_;
+}

diff -N -u -r -b -s step4_pi/simulation.hpp step5_pi_checkpoint/simulation.hpp
--- step4_pi/simulation.hpp     2016-06-15 12:27:23.359214419 -0400
+++ step5_pi_checkpoint/simulation.hpp  2016-06-15 12:27:23.975214390 -0400
@@ -27,4 +27,9 @@
     double fraction_completed() const;

     static parameters_type& define_parameters(parameters_type&);
+
+    using alps::mcbase::save;
+    using alps::mcbase::load;
+    void save(alps::hdf5::archive & ar) const;
+    void load(alps::hdf5::archive & ar);
 };
```