Serialization in Charm++

To do load balancing, we move chares to different PEs

- How do we do this for arbitrary objects?
- Charm++ has a framework for serializing data called PUP

PUP

What is PUP?

- **P**ack and **U**npack
- With PUP, chares become serializable and can be transported to memory, disk, or another processor
- Used in dynamic load balancing framework for object movement

Hello World with Chares

```
class MyChare :
public Cbase_MyChare {
   int a;
   float b;
   char c;
   entry
localArray[LOCAL_SIZE];
};
```

```
void pup(PUP::er &p) {

   p | a;
   p | b;
   p | c;
   p(localArray, LOCAL_SIZE);
}
```

Writing an Advanced PUP Routine

```
class MyChare : public Cbase_MyChare {
  int heapArraySize;
  float* heapArray;
  MyClass* pointer;
};
```

```
void pup(PUP::er &p) {
  p | headArraySize;
  if (p.isUnpacking()) {
    heapArray = new float[heapArraySize]; }
  p(heapArray, heapArraySize);
  bool isNull = !pointer;
  p | isNull;
  if (!isNull) {
    if (p.isUnpacking()) {
      pointer = new MyClass(); }
    p | *pointer; }}
```

PUP: Applicability

PUP works on:

- A simple type, e.g. `char`, `short`, `int`, `long`, `float`, or `double`
- Any object with a PUP method defined
- STL containers (`#include pup_stl.h`)
- Some others, see Section 6 of Charm++ manual for details

PUP Uses

- Moving objects for load balancing
- Marshalling user defined data types
  - When using a type you define as a parameter for an entry method
  - Type has to be serialized to go over network, uses PUP for this
  - Can add PUP to any class, doesn't have to be a chare
- Serializing for storage

Split Execution: Checkpoint Restart

- Can use to stop execution and resume later
  - The job runs for 5 hours, then will continue in new allocation another day!
- We can use PUP for this!
- Instead of migrating to another PE, just “migrate” to disk

How to Enable Split Execution

- Call to checkpoint the application is made in the main chare at a synchronization point
- `log_path` is file system path for checkpoint
- `Callback cb` called when checkpoint (or restart) is done
  - For restart, user needs to provide argument `+restart` and path of checkpoint file at runtime

```
CkCallback cb (CkIndex_Hello:SayHi(),
helloProxy);
CkStartCheckpoint(“log_path”, cb);


shell> ./charmrun hello +p4 +restart log_path
```