Abstract

In distributed computing programming, lots of large-scale data object will be involved (i.e. matrix multiplication, transposition, etc), which cannot be stored in memory on a single node. A single logical object is often needed to represent and control this large data object across a set of nodes or localities. We implement a C++ based distributed object abstraction using HPX, a C++ Standard Library for Concurrency and Parallelism. This poster will introduce an easy-to-use C++ distributed container with simple use-case, its user-friendly API of handling data transfer between localities, and its background of distributed computing and HPX.

Background

HPX (High Performance ParalleX) is a general purpose C++ runtime system for parallel and distributed applications of any scale. The following infrastructure in HPX provides support for the distributed object:

- Active Global Address Space: AGAS exposes a single uniform address space spanning all localities an application runs on.
- Component: A component is a C++ object which can be accessed remotely.
- Action: An action is a function that can be invoked remotely.

Registration Methods

- Look-ups happen on an as-needed basis
- Worst case N^2 lookups
- Currently the template’s default registration method

In every case, N lookups must be done, for each locality to find the Meta object
- Slower on startup than All-to-All but a much lower upper-bound on messages sent
- Specified as template parameter

Conclusion

- Provides an easy-to-use distributed container and offers a user-friendly API that hides communication details for user which allows easy transfer of data between localities
- Delivers high reusable code and ensures code portability
- Improves user’s programming productivity in high performance computing

Examples

```cpp
template <typename T, construction_type C = all_to_all>
class distributed_object {
  public:
    distributed_object(std::string base, T const& data,
                       std::vector<void*>& sub_localities = all_localities());
    distributed_object(distributed_object& other);`