

Math 4997-3

Lecture 18: Distributed implementation of the heat equation I

<https://www.cct.lsu.edu/~pdiehl/teaching/2019/4977/>

This work is licensed under a Creative Commons "Attribution-NonCommercial-NoDerivatives 4.0 International" license.



Reminder

Compile HPX with network support

HPX features

Update the 1D heat equation code

Scaling results

Summary

References

Reminder

Lecture 17

What you should know from last lecture

- ▶ How to use components and actions to make remote function calls

Compile HPX with network support

Parcelports [1]

To compile HPX using network support use following CMake option `-DHPX_WITH_NETWORKING=ON` and choose one of the following parcel ports:

- ▶ `HPX_WITH_PARCELPOR_T_MPI` (Message Passing Interface¹)
- ▶ `HPX_WITH_PARCELPOR_T_LIBFABRIC` (Libfabric²)
- ▶ `HPX_WITH_PARCELPOR_T_TCP` (Transmission Control Protocol)

Compile HPX with the MPI parcel port:

```
cmake -DCMAKE_BUILD_TYPE=Release \  
      -DHPX_WITH_NETWORKING=ON \  
      -DHPX_WITH_PARCELPOR_T_MPI=ON ..
```

¹<https://www.open-mpi.org/>

²<https://ofiwg.github.io/libfabric/>

Running distributed HPX applications

Using srun

```
srun -p <partition> -N <number-of-nodes> my_hpx
```

Example:

```
srun -p marvin -N 2 ./bin/hello_world
```

Using a batch job

```
#!/usr/bin/env bash
#SBATCH -o hostname_%j.out
#SBATCH -t 0-00:02
#SBATCH -p marvin
#SBATCH -N 2
```

```
srun ~/demo_hpx/bin/hello_world
```

Example:

```
sbatch example.sbatch
```

HPX features

Getting topology information³

- ▶ `hpx::find_here`
Get the global address of the locality the function is called on.
- ▶ `hpx::find_all_localities`
Get the global addresses of all available localities.
- ▶ `hpx::find_remote_localities`
Get the global addresses of all available remote localities.
- ▶ `hpx::get_num_localities`
Get the number of all available localities.
- ▶ `hpx::find_locality`
Get the global address of any locality hosting the component.
- ▶ `hpx::get_colocation_id`
Get the locality hosting the object with the given address.

³ https://stellar-group.github.io/hpx/docs/sphinx/latest/html/manual/writing_distributed_hpx_applications.html

Update the 1D heat equation code

Adding serialization functionality

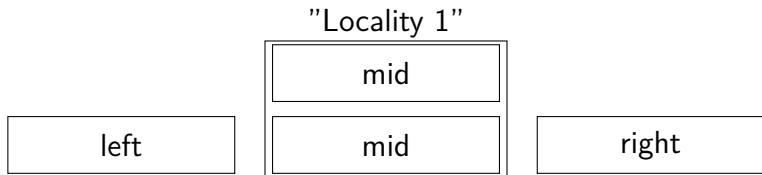
```
struct partition_data
{
private:

friend class hpx::serialization::access;

template <typename Archive>
void serialize(Archive& ar, const unsigned int version)
{
    ar & data_ & size_ & min_index_;
}

};
```

Reducing the overhead of copying I



```
struct partition_server
: hpx::components::component_base<partition_server>
{

enum partition_type
{
    left_partition, middle_partition, right_partition
};

};
```

Reducing the overhead of copying II

```
partition_data get_data(partition_type t) const
{
    switch (t)
    {
        case left_partition:
            return partition_data(data_, data_.size()-1);

        case middle_partition:
            break;

        case right_partition:
            return partition_data(data_, 0);

        default:
            HPX_ASSERT(false);
            break;
    }

    return data_;
}
```

Reducing the overhead of copying III

```
struct partition : hpx::components::client_base<
    partition, partition_server>
{
    //We pass no the type of the partition to the action
    // to avoid copying the mid partition as it is on
    // the same locality
    hpx::future<partition_data> get_data(
        partition_server::partition_type t) const
    {
        partition_server::get_data_action act;
        return hpx::async(act, get_id(), t);
    }
};
```

Reducing the overhead of copying IIII

```
return dataflow(
    hpx::launch::async,
    unwrapping(
        [left, middle, right](partition_data const& l,
            partition_data const& m,
            partition_data const& r)
        {
            HPX_UNUSED(left);
            HPX_UNUSED(right);

            return partition(middle.get_id(),
                heat_part_data(l, m, r));
        }
    ),
    left.get_data(partition_server::left_partition),
    middle.get_data(partition_server::middle_partition),
    right.get_data(partition_server::right_partition)
);
```

Distributing the work to the localities

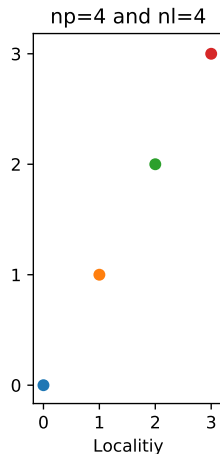
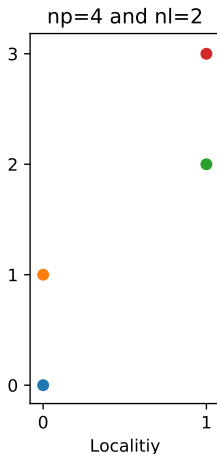
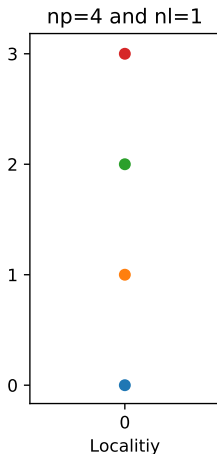
```
// Find all available localities
std::vector<hpx::id_type> localities =
    hpx::find_all_localities();
// Determine the number of localities
std::size_t nl = localities.size();

// Generate the partition on the localities
// Note before we had hpx::find_here there
for (std::size_t i = 0; i != np; ++i)
    U[0][i]
        = partition(localities[locidx(i, np, nl)],
            nx, double(i));
```

We use `locidx` to decide on which locality the partition is generated.

Define the locality

```
std::size_t locidx(std::size_t i, std::size_t np,  
                  std::size_t nl)  
{  
    return i / (np/nl);  
}
```



Scaling results

Configuration file

```
#!/usr/bin/env bash

#SBATCH -o hostname_%j.out
#SBATCH -t 00:25:00
#SBATCH -p medusa
#SBATCH -D /home/pdiehl/Compile/hpx-1.3.0/build/bin/
export LD_LIBRARY_PATH
        =$LD_LIBRARY_PATH:
        /home/pdiehl/Compile/hpx-1.3.0/build/lib

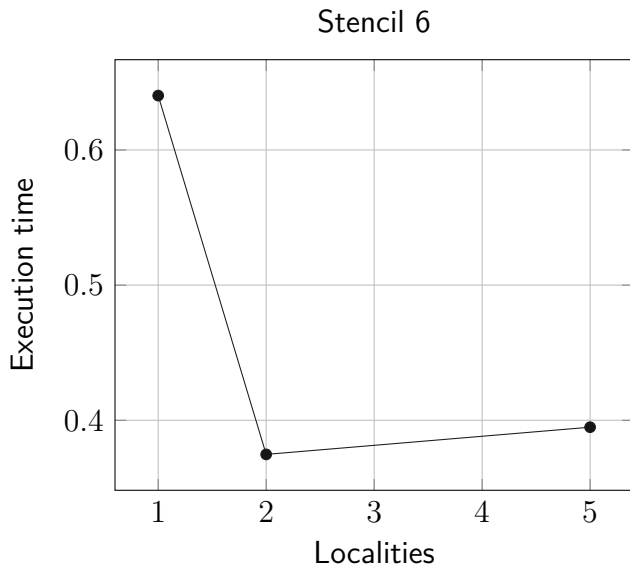
module load gcc/8.2.0 boost/1.69.0-gcc8.2.0-release
        mpi/openmpi-x86_64

srun 1d_stencil_6 --nx=1000000 --np=10
```

Running

```
sbatch -N 1,2,3,4,5 stencil.sbatch
```

Distributed scaling



Summary

Summary

After this lecture, you should know

- ▶ How to compile HPX using networking
- ▶ Receiving topology information

References

References I



Hartmut Kaiser, Maciek Brodowicz, and Thomas Sterling.
Parallex an advanced parallel execution model for
scaling-impaired applications.

*In 2009 International Conference on Parallel Processing
Workshops*, pages 394–401. IEEE, 2009.