Math 4997-3

Lecture 19: Distributed implementation of the heat equation II

Patrick Diehl

https://www.cct.lsu.edu/~pdiehl/teaching/2021/4997/

Notes

Notes

Notes

Reminder

Notes

Lecture 18

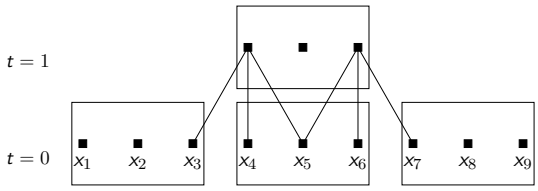
What you should know from last lecture

- How to compile HPX using networking
- Receiving topology information

Improving sending data

Notes

Exchanging boundary data



In the previous example we exchanged the whole left and right partition to update the temperatures of the partition in the middle. However, we only need one value of the neighboring partitions to update the temperature for x_4 and x_6.

Notes

Updating the partition I

```
// Create a partition only having the left or
// right boundary element
partition_data(partition_data const& base,
  std::size_t min_index)
  : data_(base.data_.data()+min_index, 1,
        buffer_type::reference),
        size_(base.size()),
        min_index_(min_index)
{
        HPX_ASSERT(min_index < base.size());
}

double& operator[](std::size_t idx) {
        return data_[index(idx)]; }

double operator[](std::size_t idx) const {
        return data_[index(idx)]; }
```

Notes

Updating the partition II

```
private:
std::size_t index(std::size_t idx) const
{
        HPX_ASSERT(idx >= min_index_ && idx < size_);
    return idx - min_index_;
}
```

Now, we have to make use of sending smaller partitions to the node containing the middle partitions. We will use the previously introduced partition type.

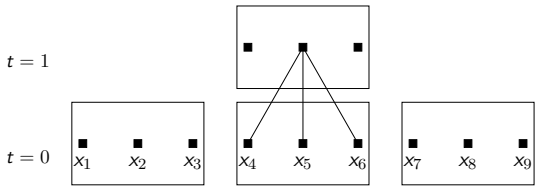
Notes

Updating the partition server

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

    case middle_partition:
      break;

    case right_partition: // First element
      return partition_data(data_, 0);

    default:
       HPX_ASSERT(false);
       break;
  }
  return data_;
}
```

Notes

Improve the computation of the middle partition



We only need the left and right partition to compute the updated temperature x_4 and x_6, but we can compute the temperature for x_5 without having the partitions Thus, we could update the values in the mid of the partition, while waiting to receive the partitions.

Notes

Update the stepper: Compute local

```
hpx::shared_future<partition_data> middle_data =
  middle.get_data(partition_server::middle_partition);

hpx::future<partition_data> next_middle =
  middle_data.then(
  unwrapping(
    [middle](partition_data const& m) -> partition_data
    {
       HPX_UNUSED(middle);

       // All local operations are performed once the
       // middle data of the previous time step becomes
       // available.
       std::size_t size = m.size();
       partition_data next(size);
       for (std::size_t i = 1; i != size-1; ++i)
          next[i] = heat(m[i-1], m[i], m[i+1]);
       return next;
}));
```

Notes

Update the stepper: Compute the boundary values

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

        // Calculate the missing boundary elements once the
        // corresponding data has become available.
        std::size_t size = m.size();
        next[0] = heat(l[size-1], m[0], m[1]);
        next[size-1] = heat(m[size-2],
                        m[size-1], r[0]);
```

Notes

Update the stepper: Compute the boundary values

```
    // The new partition_data will be allocated on the
    // same locality as 'middle'.
            return partition(middle.get_id(), next);
            }),
                    std::move(next_middle),
                    left.get_data(
                      partition_server::left_partition),
                    middle_data,
                    right.get_data(
                      partition_server::right_partition)
            );
        }
```

Notes

Scaling results

Notes

Configuration file

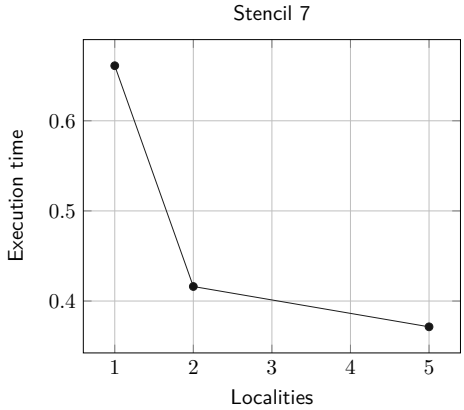
```
#!/usr/bin/env bash

#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_7 --nx=1000000 --np=10
```

Running

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

Notes

Distributed scaling



Notes

Summary

Notes

Summary

Notes

After this lecture, you should know

- Running distributed HPX applications
- Using the slurm environment and modules on clusters

Notes

Notes