Progress of Thesis Work

and discussion on current/next steps

Chengxin Ma

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

October 10, 2019

Table of Contents

Table of Contents

Lessons Learned from LLNL Tutorial[1]

- Automatic vs. Manual Parallelization
- Understand the Problem and the Program
- **Partitioning**
- **Communication**
- Synchronization
- Data Dependencies
- Load Balancing
- **Granularity**
- I/O

The aspects in **bold** format are major concerns in my thesis project.

[1] `https://computing.llnl.gov/tutorials/parallel_comp/`

Automatic vs. Manual Parallelization

Manual.

More control over the problem to be solved.

Understand the Problem and the Program

This slide needs to be discussed.

- Hotspots (where most of the real work is being done)
  - Sorting
- Bottlenecks
  - Disk I/O (so we use Arrow stuff)
  - Network/communication
    - Pattern: all-to-all communication
  - What else?

Partitioning

- **Domain decomposition** (we go this way due to the nature of the problem)
- Functional decomposition

Communication

- Communication overhead
  - Machine cycles and resources that could be used for computation are instead used to package and transmit data. (Using RDMA, it is possible to move the workload of communication from CPU and OS to, e.g., NIC.)
  - Communications frequently require some type of synchronization between tasks, which can result in tasks spending time "waiting" instead of doing work. (This underlines the importance of scheduling.)
  - Competing communication traffic can saturate the available network bandwidth, further aggravating performance problems. (ditto.)
- Latency and bandwidth
  - We desire low latency and high bandwidth. (RDMA offers such features.)

Communication

- **Visibility of communications**
  - With the **Message Passing Model** (we choose this option), communications are explicit and generally quite visible and under the control of the programmer.
  - With the Data Parallel Model [2], communications often occur transparently to the programmer, particularly on distributed memory architectures. The programmer may not even be able to know exactly how inter-task communications are being accomplished.
- Synchronous vs. **asynchronous communication** (we go for asynchronous communication)
- Scope of communication: all-to-all

[2] May also be referred to as the Partitioned Global Address Space (PGAS) model. https://computing.llnl.gov/tutorials/parallel_comp/#ModelsData

Synchronization

To be considered when it goes to the next phase of design.

Data Dependencies

To be considered when it goes to the next phase of design.

Load Balancing

To be considered when it goes to the next phase of design.
If data partitioning is done well, load might be almost balanced.

Granularity

Qualitative measure of the ratio of computation to communication.

- Fine-grain Parallelism: sort part of the data locally, send to destination, repeat until all data is processed
- Coarse-grain Parallelism: sort all the data locally, send to destination

(The final design might be fine-grain palleelism, but we can start from the coarse-grain parellelism and iteratively improve performance.)

I/O

Use in-memory technologies instead.
Not the concern in my thesis project.

Table of Contents

Underlying Hardware Architecture

This slide needs to be discussed.

Nodes connected by network (referred to as Hybrid Distributed-Shared Memory in the LLNL tutorial)
Interconnection technologies:

- InfiniBand
- 10 Gigabit Ethernet

Some [3] says either of them is OK. How to identify if communication bandwidth/latency is indeed the bottleneck?

[3] https://www.hpcadvisorycouncil.com/pdf/IB_and_10GigE_in_HPC.pdf

Thoughts on RDMA

This slide needs to be discussed.

A rough layering:

User application (sorting SAM)
User level libraries, such as MPI
Lower level libraries which implements RDMA
Hardware that supports RDMA

My opinion on RDMA: from the programmer's perspective, usage of RDMA might be implicit. "RDMA programming interfaces are often not designed to be used by end-users directly". [4]
I guess the major part of my work would be **using** the user level libraries to write the user applications, instead of **writing** the user level libraries.

[4] https://htor.inf.ethz.ch/blog/index.php/2016/05/15/what-are-the-real-differences-between-rdma-infiniband-rma-and-pgas/

Parallel Programming Model

Message Passing Model

Frameworks that Meet the Requirements

This slide needs to be discussed.

Based on the discussion above, so far the requirements are:

- Support for InfiniBand (RDMA) or 10 Gigabit Ethernet
- Using the Message Passing Model (for explicitly controlling communication)

The following frameworks meet the these requirements:

- MPI [5] over InfiniBand [6]
- What else?

Alternatively, we can build up our own framework starting from scratch. The benefit is the high degree of freedom to customize the system.

---

[5] MPI is interconnect independent: `https://stackoverflow.com/a/30400654/5723556`

[6] `https://www.open-mpi.org/papers/workshop-2006/thu_01_mpi_on_infiniband.pdf`

Table of Contents

About the Ray Project

I will take a look after I get more knowledge about Plasma.

Table of Contents

Next steps

Comments and suggestions?