Parallel Computing

Majid Almeshari
John Conklin

Outline

- **The Challenge**
- **Available Parallelization Resources**
- **Status of Parallelization**
- **Plan & Next Step**

The Challenge

1. Sigma Point Filtering is used as our non-linear estimation filter

- Carrying this out on a 2-second time step is computationally intensive and requires handling a huge set of data

2. The data set is huge ~ 1 Terabyte

- Resources such as RAM may not be sufficient to carry out the computation tasks efficiently. Swapping to the disk delays the processing as well

In a nutshell, A computationally intensive algorithm is applied on a huge set of data

- Testing takes a long time which delays the verification of the correctness of the algorithms

Available Parallelization Resources

- **Hardware**
  - **2 Computer Clusters** *(Namely: Regelation and Nivation)*
    - » **Regelation : 44 64-bit-CPUs**

      64-bit enables us to address a memory space beyond 4 GB
    - » **Nivation: 150 32-bit-CPUs**
    - » **Both clusters run linux**
- **Software**
  - **Matlab**
  - **MatlabMPI**
    - » A toolbox for parallel computing using Matlab
  - **Other parallel computing toolboxes have been investigated as alternatives when MatlabMPI proves to be inefficient**

Status - 1

- **We managed to run a first test using MatlabMPI on the cluster**
  - **Master-Slave architecture**
  - **Single-Instruction Multiple-Data (SIMD) parallel processing style**
- **Results**
  - **Pure computation time was improved**
  - **However, Inter-Processor communication overhead took about 90% of the processing time. This is due to**
    - Large message size, 5 MB in average
    - Total number of messages can go up to 16,000 (~4000 orbit X 4 gyros)
    - That is ~80 GB of reading/writing to the disk + network

Status - 2

- **This required a re-arrangement of the processing procedure to:**
  - Minimize communication time
  - Distribute the work non-uniformly among processors
- **At the same time, we are looking deeper inside the code to improve efficiency (computation, RAM). Focus is on:**
  - Modules that take most of the computation time (e.g. building the Jacobian, 80% of the time)
  - Modules that consume most of the RAM

Plan

- **We are working closely with the development team to understand the code better** *(until the end of June)*
  - Further understanding of Sigma-Point Kalman Filter will help us parallelize it better
  - Knowing the dependencies among modules will help us minimize inter-processor communication time
- **We are running small tests whenever the code changes to see and record any improvement** *(until the end of June)*

- **On the long run, We plan to:**
  - **Design a parallel framework where we assign which modules to run where** *(by the end of July)*
  - **Implement and test this framework incrementally** *(Complete by the end of September)*