Parallel Processing

Majid AlMeshari
John W. Conklin
Outline

• Challenge
• Requirements
• Resources
• Approach
• Status
• Tools for Processing
Challenge

A computationally intensive algorithm is applied on a huge set of data. Verification of filter’s correctness takes a long time and hinders progress.
Requirements

- Achieve a speed-up between 5-10x over serial version
- Minimize parallelization overhead
- Minimize time to parallelize new releases of code
- Achieve reasonable scalability
Resources

• Hardware
  – Computer Cluster (Regelation), a 44 64-bit CPUs
    » 64-bit enables us to address a memory space beyond 4 GB
    » Using this cluster because:
      (a) likely will not need more than 44 processors
      (b) same platform as our Linux boxes

• Software
  – MatlabMPI
    » Matlab parallel computing toolbox created by MIT Lincoln Lab
    » Eliminates the need to port our code into another language
Approach - Serial Code Structure

Initialization

Loop over iterations

Loop Gyros

Loop over Segments

Loop over orbits

Build $h(x)$ (the right hand side)

Build Jacobian, $J(x)$

Load data ($Z$), perform fit

Computationally Intensive Components

(Parallelize!!!)

Bayesian estimation, display results
Approach - Parallel Code Structure

Initialization

Loop over iterations

Loop Gyros

Loop over Segments

Loop over orbits

Build $h(x)$

Build Jacobian, $J(x)$

Distribute $J(x)$, all nodes

Loop over partial orbits

Load data ($Z$), perform fit

Send information matrices to master & combine

In parallel, split up by columns of $J(x)$

Custom data distribution code required

(Overhead)

In parallel, split up by orbits

Bayesian estimation, display results
Code in Action

Initialization
- Compute h and Partial J
- Sync Point
- Get Complete J
- Perform Fit
- Sync Point
- Combine Info Matrix

Bayesian Estimation

Another Iteration?

Split by Columns of J
- Go for next Gyro, Segment
- Split by Rows of J
- Go for another Iteration

Go for next iteration

Proc 1  Proc 2  Proc 3  . . . . . .  Proc n

Initialization

Compute h and Partial J

Sync Point
Get Complete J

Perform Fit

Sync Point
Combine Info Matrix

Bayesian Estimation

Another Iteration?
Parallelization Techniques - 1

- Minimize inter-processor communication
  - Use MatlabMPI for one-to-one communication only
  - Use a custom approach for one-to-many communication
Parallelization Techniques - 2

- Balance processing load among processors

Load per processor = \[ \sum \text{length(reducedVector}_k\text{)} \times (\text{computationWeight}_k + \text{diskIOWeight}) \]

\[ k: \{RT, Tel, Cg, TFM, S_0\} \]
Parallelization Techniques - 3

• Minimize memory footprint of slave code
  – Tightly manage memory to prevent swapping to disk, or “out of memory” errors
  – We managed to save ~40% of RAM per processor and never seen “out of memory” since
Parallelization Techniques - 4

• Find contention points
  – Cache what can be cached
  – Optimize what can be optimized

Example: Compute h Optimization
Status – Current Speed up

- Sep '09
- May '10
- July '10

Number of Processors vs. Speedup
Tools for Processing - 1

• Run Manager/Scheduler/Version Controller
  – Built by Ahmad Aljadaan
  – Given a batch of options files, it schedules batches of runs, serial or parallel, on cluster
  – Facilitates version control by creating a unique directory for each run with its options file, code and results
  – Used with large number of test cases
  – Helped us schedule 1048+ runs since January 2010
    » and counting…
Tools for Processing - 2

- **Result Viewer**
  - Built by Ahmad Aljadaan
  - Allows searching for runs, displays results, plots related charts
  - Facilitates comparison of results
Questions