# Enhanced File Interoperability with Parallel MPI File I/O in Image Processing

D.A.L. Piriyakumar, P. Levi
R. Rabenseifner
University of Stuttgart
Germany



## Image Processing

- Image Matching
  - Matching a model with images
- Computationally Intensive
- Lot of Applications
  - Robotics, Satellite Imagery, Medical Imaging
- Human being Recognition
  - Surveillance, Industrial Environments


# Image Matching Problem

- Given Set of Models
- Sequence of Images
- Find Positions of Human beings in the Images
- Models and Images with Corners
- Appropriate Matching Measure







# Hausdorff Measure

- Hausdorff Method very General
- Given Two Sets of Points
- Find the Maximum Misalignment
- Corners of Models and Images
  - Form Point Sets
- Hausdorff Measure = Matching Measure
- Proper Threshold


# Parallel Algorithm

- 3 Methods to parallelize
  - Take each Image by a Processor, match all Models
  - Take one Image by all Processors, match subset of Models
  - Take subset of Image by a Processor, match all Models
- Second Method good for Tracking
- Crucial File Reading
  - Images, Models in ASCII form Integers






# File Interoperability in MPI-2

- MPI Provides Parallel I/O
  - MPI_File_read_all or MPI_File_read
- General I/O Overheads
- Problem in File Reading
  - File in ASCII form Integers
  - Not directly Readable by MPI in Cray T3E
- File Interoperability needed in MPI-2


# Data Representations in MPI-2

- MPI-2 has 3 Data Representations
  - Native – as in Memory
  - Internal – Implementer Dependent
  - External 32 – Binary Representation
  - All Not useful to read ASCII form Integers
  - With User-defined data
    - Must have a well-defined number of bytes
    - No chance to read as Integers may be in different digits






# Reading ASCII form Integer File with MPI I/O

- 3 Strategies
  - Normal File Reading with fscanf
    - No Need for Conversion, but No MPI I/O also
  - Off-line Conversion
    - Convert to Native Format off-line
  - Runtime Conversion
    - Reading as Char, then Conversion to Integer
- 3 Strategies Implemented and Analyzed
- Further Optimization Possible ?

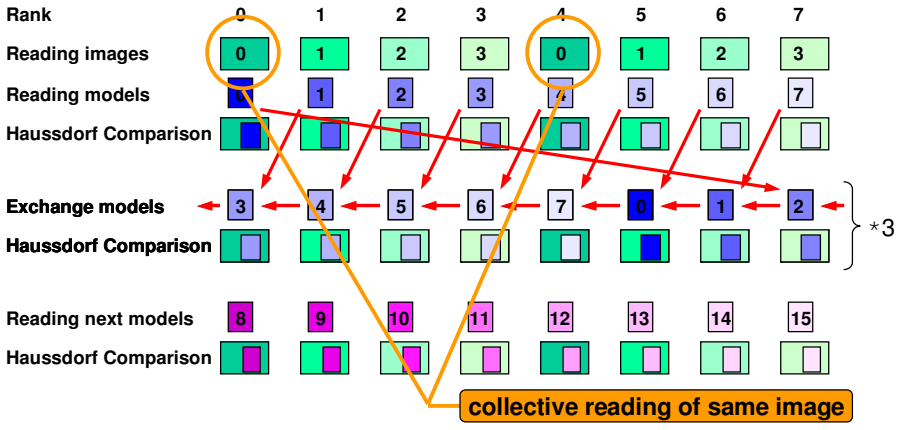

# Optimization of Parallel I/O

- Two Chances for Optimization
  - Reuse already read Image Data File
  - Reuse already read Model Data File
- Collective Reading of Image Data File
- Caching latest read Model Data File
- If more than one Image to be read in Parallel
  - A New Optimization Scheme

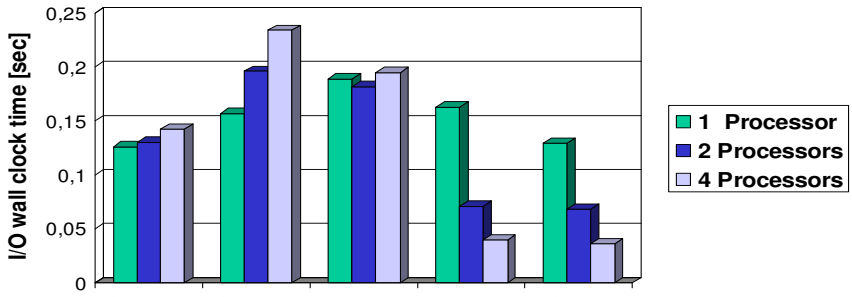





# Optimized Parallelization Scheme




# I/O Time per Process




| Parallelization: | Original | Original | Original | **Optimized** | **Optimized** |
|---|---|---|---|---|---|
| File I/O: | fscanf | MPI-I/O | MPI-I/O | MPI-I/O | fscanf |
| Conversion: | online | online | **off**line | online | online |
| I/O entities: | Integers | whole file | 3*Int + 2*array | whole file | Integers |






# Experimental Data

- 4 Images and Models
- On Cray T3E with 1 or 2 or 4 Processors
- MPI_Wtime() used for Timing Measurements
- MPI_File_read used for Reading Files
  - Individual Characters
  - File of Characters to minimize High Latency
- Larger Files – Larger I/O Timings


# Analysis of Results

- MPI I/O can be used for ASCII Files
- Large chunks to be read to minimize Latency
- Optimize Image Processing Methods
  - Than MPI Library
- For Small and Medium size Files use fscanf
  - Than Message Passing or Broadcasting
  - Than using Collective MPI I/O






# Conclusions

- Parallel Image Matching better with MPI
- Parallel MPI I/O must be carefully optimized
- 3 strategies suggested for Reading ASCII Files
- Thus, Enhanced File Interoperability in MPI
- Accurate Positions of Human beings found
- Parallelization of the Application preferred
- File Format Conversion can be solved efficiently