SINBAD Consortium 2010

Parallelizing Operations with Ease Using Parallel SPOT

Nameet Kumar

SLIM
University of British Columbia

Overview

- Motivation
- Parallel Computing Toolbox - Matlab
- Parallel SPOT
- Examples & Demo

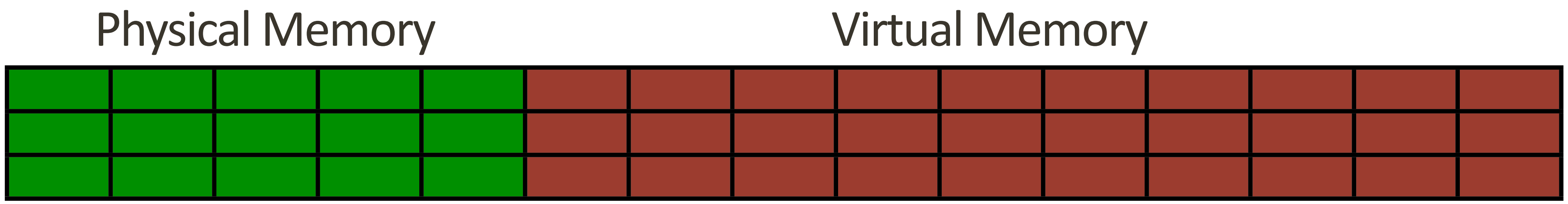
Motivation

Limitations of a single system

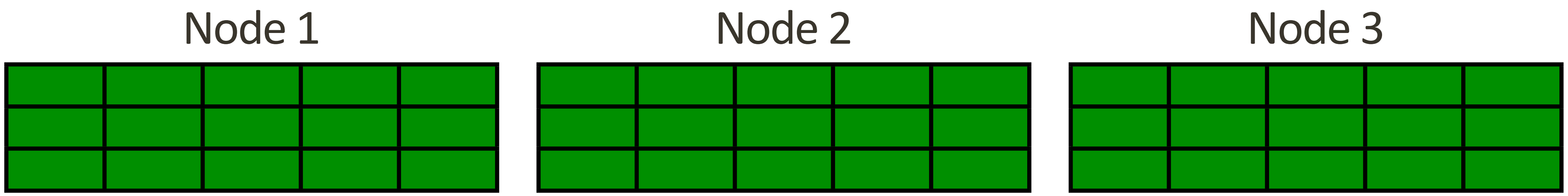
- Memory -- problem size
- Processing power -- speed

Increasing Problem Size

- Real data is often too large to fit in memory on a single computer



- But can be split among several computers



Improving Speed

- Data-intensive operations can often be drastically sped up by distributing the work

- Situations ideal for distribution are called Embarrassingly Parallel
- Still other operations can be parallelized with some consideration

Embarrassingly Parallel

Operations you can split with little to no communication

$$\begin{bmatrix} 0 & 1 & 0 & - & - & - & - & - & - \\ 1 & 0 & 1 & - & - & - & - & - & - \\ 1 & 0 & 0 & - & - & - & - & - & - \\ - & - & - & 8 & 4 & 4 & - & - & - \\ - & - & - & 8 & 3 & 9 & - & - & - \\ - & - & - & 1 & 8 & 2 & - & - & - \\ - & - & - & - & - & - & 1 & 0 & 0 \\ - & - & - & - & - & - & 0 & 1 & 0 \\ - & - & - & - & - & - & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 2 \\ 1 \\ 1 \\ 8 \\ 6 \\ 3 \\ 5 \\ 4 \\ 0 \end{bmatrix}$$

```
A = opBinary( 3 );      B = randi( 9, 3 );      C = opEye( 3 );
BD = opBlockDiag( A, B, C );          BD ∗ x;
```

Embarrassingly Parallel

Operations you can split with little to no communication

$$\begin{bmatrix} 0 & 1 & 0 & - & - & - & - & - & - \\ 1 & 0 & 1 & - & - & - & - & - & - \\ 1 & 0 & 0 & 8 & 4 & 4 & - & - & - \\ - & - & - & 8 & 3 & 9 & - & - & - \\ - & - & - & 1 & 8 & 2 & 1 & 0 & 0 \\ - & - & - & - & - & - & 0 & 1 & 0 \\ - & - & - & - & - & - & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 1 \\ 1 \\ 8 \\ 6 \\ 3 \\ 5 \\ 4 \\ 0 \end{bmatrix}$$

A = opBinary(3);   B = randi(9, 3);   C = opEye(3);

BD = opBlockDiag(A, B, C, **1);**   BD $*$ x;

Non-Trivial Parallelizing

Requires extensive communication and planning

$$y = (A \otimes B) \times \text{vec}(X)$$

Simplified using the equality:

$$(A \otimes B) \times \text{vec}(X) = \text{vec}(B \times X \times A^T) = \text{vec}((A \times (B \times X)^T)^T)$$

Parallel Matlab

- Parallel Computing Toolbox provides constructs and data types for parallel programming

- Constructs:
  - spmd - Single Program Multiple Data
  - parfor - Parallel For loop
- Data Types:
  - Distributed/Codistributed - matrix/cell

Parallel Matlab

```
BD = distributed.cell( 1, 3 );
BD{1} = A;      BD{2} = B;      BD{3} = C;
y = cell( 3, 1 );
spmd
   block = getLocalPart( BD );
   y{ labindex } =  block * x( 1 + (labindex-1)*3 : labindex*3 );
end
y = cell2mat( y );
```

Parallel SPOT

- Is object oriented and thus **intuitive** to use
- Abstracts away the translation to parallel code
- Is written in MatLab
  - ➡interfaces directly with Java
  - ➡works well with C/C++ and Fortran
- Is an extension to SPOT
  - ➡Has the same framework
  - ➡Compatible with SPOT operators
  - ➡Inherits/overloads all the superclass functions

Parallel Block Diagonal

$$
\begin{bmatrix}
0 & 1 & 0 & - & - & - & - & - & - \\
1 & 0 & 1 & - & - & - & - & - & - \\
1 & 0 & 0 & 8 & 4 & 4 & - & - & - \\
- & - & - & 8 & 3 & 9 & - & - & - \\
- & - & - & 1 & 8 & 2 & 1 & 0 & 0 \\
- & - & - & - & - & - & 0 & 1 & 0 \\
- & - & - & - & - & - & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
2 \\ 1 \\ 1 \\ 8 \\ 6 \\ 3 \\ 5 \\ 4 \\ 0
\end{bmatrix}
$$

```
A = opBinary( 3 );      B = randi( 9, 3 );      C = opEye( 3 );
BD = opBlockDiag( A, B, C, 1 );
BD * x;
```

Parallel Block Diagonal

$$\begin{bmatrix} 0 & 1 & 0 & - & - & - & - & - & - \\ 1 & 0 & 1 & - & - & - & - & - & - \\ 1 & 0 & 0 & 8 & 4 & 4 & - & - & - \\ - & - & - & 8 & 3 & 9 & - & - & - \\ - & - & - & 1 & 8 & 2 & 1 & 0 & 0 \\ - & - & - & - & - & - & 0 & 1 & 0 \\ - & - & - & - & - & - & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 1 \\ 8 \\ 6 \\ 3 \\ 5 \\ 4 \\ 0 \end{bmatrix}$$

A = opBinary(3);   B = randi(9, 3);   C = opEye(3);

BD = ~~opBlockDiag(A, B, C, 1);~~

BD $*$ x;

Parallel Block Diagonal

$$\begin{bmatrix} 0 & 1 & 0 & - & - & - & - & - & - \\ 1 & 0 & 1 & - & - & - & - & - & - \\ 1 & 0 & 0 & 8 & 4 & 4 & - & - & - \\ - & - & - & 8 & 3 & 9 & - & - & - \\ - & - & - & 1 & 8 & 2 & 1 & 0 & 0 \\ - & - & - & - & - & - & 0 & 1 & 0 \\ - & - & - & - & - & - & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 1 \\ 1 \\ 8 \\ 6 \\ 3 \\ 5 \\ 4 \\ 0 \end{bmatrix}$$

A = opBinary(3); B = randi(9, 3); C = opEye(3);

BD = ~~opBlockDiag(A, B, C, 1);~~ oppBlockDiag(A, B, C, 1);

BD $*$ x;

Parallel Kronecker

$$(A \otimes B) \times \mathrm{vec}(X) = \mathrm{vec}((A \times (B \times X)^T)^T)$$

- A serial operation by nature
- Break down to a sequence of parallel operations

Future of Parallel SPOT

- Contain parallel implementations of all SPOT operators
- Parallelism should be a ‘switch’ in SPOT
- Incorporate load balancing at the highest level

- A SPOT friendly distributed container for data to encapsulate:
  ✓Parallel I/O
  ✓Meta-data - header info, original dimensions
  ✓Shape - current shape, smart reshaping methods

Acknowledgements

- Henryk Modzelewski

SINBAD

NSERC CRSNG

This work was in part financially supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (22R81254) and the Collaborative Research and Development Grant DNOISE II (375142-08). This research was carried out as part of the SINBAD II project with support from the following organizations: BG Group, BP, Chevron, ConocoPhillips, Petrobras, Total SA, and WesternGeco.