

Data Splitting for Parallel Linear Algebra Monte Carlo Methods

Christian Weihrauch

`c.weihrauch@reading.ac.uk`

Centre for Advanced Computing and Emerging Technologies
The University of Reading

Lunchtime Seminar - 7th March 2007

- 1 Introduction to Monte Carlo Method
 - Monte Carlo Method
 - Markov Chains
 - Quasi Monte Carlo Method
 - Example I
 - Example II
- 2 Monte Carlo Methods for Linear Algebra
 - Problem to Solve
 - Algorithm
 - Parallel Approach
- 3 Data Splitting for Parallel Linear Algebra Monte Carlo
 - Introduction
 - Motivation
 - Data Splitting
 - Communication
 - Implementation
 - Results
- 4 Conclusion

1 Introduction to Monte Carlo Method

Monte Carlo Method

Markov Chains

Quasi Monte Carlo Method

Example I

Example II

2 Monte Carlo Methods for Linear Algebra

Problem to Solve

Algorithm

Parallel Approach

3 Data Splitting for Parallel Linear Algebra Monte Carlo

Introduction

Motivation

Data Splitting

Communication

Implementation

Results

4 Conclusion

Monte Carlo Method

- Monte Carlo method (MC) is a numerical method using random variables for solving mathematical problems
- First real use in Manhattan project¹
- Theoretical foundations known since late 19th century
- Computer needed to generate sufficient number of random variables

¹by N.Metropolis, S.Ulam and J.v.Neumann

Monte Carlo Method

Motivation:

- Many scientific problems have high complexity
- Idea is to find models with lower complexity using MC
- Parallel MC bridges performance gap of super computers

Can be used for two different kind of mathematical problems:

- ① Problems depending on random factors
- ② Problems not depending on random factors for which artificial models can be found to simulate the problem

Monte Carlo Method

Used in many different scientific areas:

- Physics
- Medicine: image analysis
- Archeology: radiocarbon dating
- Maths: finding eigenvalue
- Maths: matrix inversion, Solving Systems of Linear Algebraic Equation (SLAE)
- Economics: SLAE, assessing the level of exposure to risk for portfolio management

Monte Carlo Idea

- Wish to estimate quantity a
- Define random variable ξ
- Where ξ has the mathematical expectation a
- Take N independent realisations ξ_i of ξ
- Then $\bar{\xi} = \frac{1}{N} \sum_{i=1}^N \xi_i$
- And $\bar{\xi} \approx a$

Markov Chains

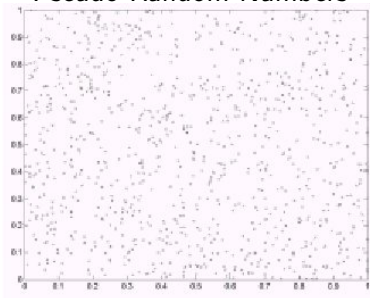
- A sequence of random variables
- Having the property that, given the present, the future is conditionally independent of the past
- A simple random walk is an example of a Markov chain
- Markov chains are used in linear algebra, e.g. random walks used to jump within the matrix

Quasi Monte Carlo Method

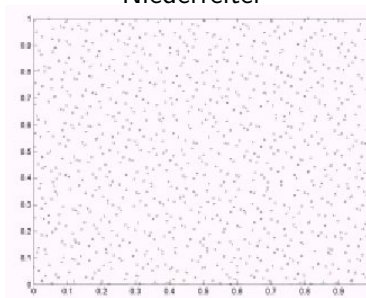
- Quasi random numbers are well distributed
- Replace standard pseudo random number generator...
- ...with Halton, Sobol or Niederreiter sequence generators
- Quasi Monte Carlo known to be faster
- Reason I: for some problems Quasi Monte Carlo Methods converge faster
- Reason II: implementations of pseudo random number generators trying hard to introduce randomness by using complex algorithms and exploiting hardware specific anomalies

Random Number Distribution

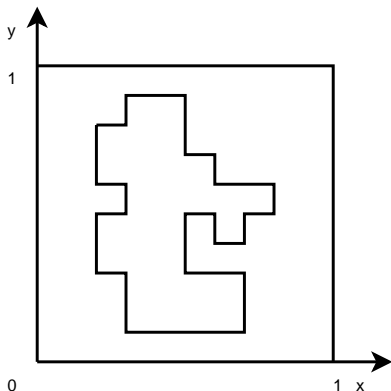
Pseudo Random Numbers



Niederreiter

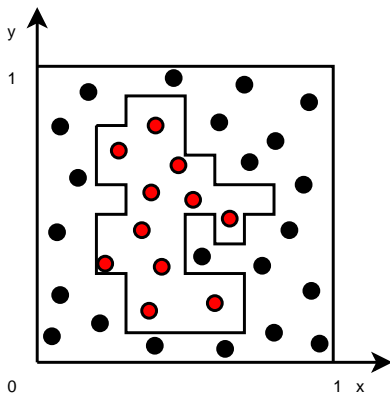


Monte Carlo Method - Example I



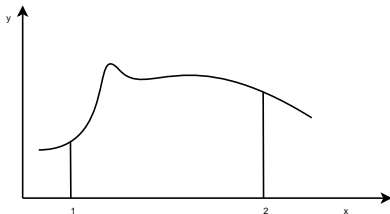
- Area covered by a complex figure
- Figure is in a unit square
- Difficult to compute the area for dimensions > 2

Monte Carlo Method - Example I



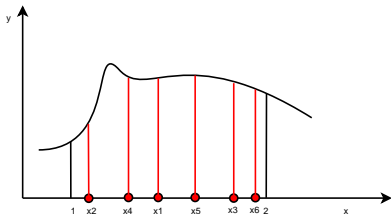
- Generate $N = 33$ random points in the unit square
- $N' = 11$ of these points are inside the figure
- Area $\approx \frac{N'}{N} = \frac{11}{33} = 0.33$; real area covered is $\frac{29}{100} = 0.29$

Monte Carlo Method - Example II



- Integral of a function $f(x)$
- Interval between 1 and 2

Monte Carlo Method - Example II



- Generate six random points $x_1 \dots x_6$ within the interval
- Covered area = $\frac{f(x_1)+f(x_2)+\dots+f(x_6)}{6} \times (2 - 1)$

- 1 Introduction to Monte Carlo Method
 - Monte Carlo Method
 - Markov Chains
 - Quasi Monte Carlo Method
 - Example I
 - Example II
- 2 Monte Carlo Methods for Linear Algebra
 - Problem to Solve
 - Algorithm
 - Parallel Approach
- 3 Data Splitting for Parallel Linear Algebra Monte Carlo
 - Introduction
 - Motivation
 - Data Splitting
 - Communication
 - Implementation
 - Results
- 4 Conclusion

Scientific and Engineering Problems

Many scientific problems revolve around:

- inverting a real $n \times n$ matrix (MI)
 - Given A
 - Find A^{-1}
- solving a system of linear algebraic equations (SLAE)
 - Given A and b
 - Solve, for x , $Ax = b$
 - Or find A^{-1} and calculate $x = A^{-1}b$

Solving SLAEs

Traditional Methods:

- Gaussian elimination
- Gauss-Jordan
- Both take $O(n^3)$ steps

Time prohibitive if:

- a large problem
- or a real time solution is required

Reason for Using Monte Carlo

$O(NT)$ steps to find an element of the:

- matrix inverse A^{-1}
- solution vector x

where:

- N number of Markov Chains
- T length of Markov Chains

Independent of n - size of matrix or problem

Algorithms can be efficiently parallelised

Transition Probability Matrix

For the Monte Carlo computations a probability matrix P is required. Two different methods of generating this matrix are common:

- Uniform Method (UM) using uniform transition probability (UM) $p_{ij} = \frac{1}{n}$
- Monte Carlo Almost Optimal (MAO) with $p_{ij} = \frac{|\alpha_{ij}|}{\sum_{k=1}^n |\alpha_{ik}|}$, where $i, j = 1, 2, \dots, n$.

Standard MC matrix inversion

Step 1. Read in matrix B , the matrix to be inverted

I: Input matrix B , parameters ε and δ

Step 2. Calculate intermediate matrices (B_1, B_2)

I: Split $B = B_1 - B_2$, where $B_1 = \text{diag}(B)$ and $B_2 = B_1 - B$

Step 3. Calculate matrix A and $\|A\|$

I: Compute matrix $A = B_1^{-1}B_2$

2: Compute $\|A\|$ and the number of Markov Chains $N = \left(\frac{0.6745}{\varepsilon(1-\|A\|)}\right)^2$

Step 4. Calculate matrix P

I: Compute the probability matrix, P , where $p_{ij} = \frac{|a_{ij}|}{\sum_{k=1}^n |a_{ik}|}$

Step 5. Calculate matrix M , by MC on A and P

I: For $i = 1$ to n

1.1: For $j = 1$ to N

Markov Chain Monte Carlo Computation

1.1.1: Set $W_0 = 1$, $point = i$ and $SUM[k] = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$

1.1.2: Select a $nextpoint$, based on the transition probabilities in P , such that $A[point][nextpoint] \neq 0$

1.1.3: Compute $W_j = W_{j-1} \frac{A[point][nextpoint]}{P[point][nextpoint]}$

1.1.4: Set $SUM[nextpoint] = SUM[nextpoint] + W_j$

1.1.5: If $|W_j| > \delta$ set $point = nextpoint$ and goto 1.1.2

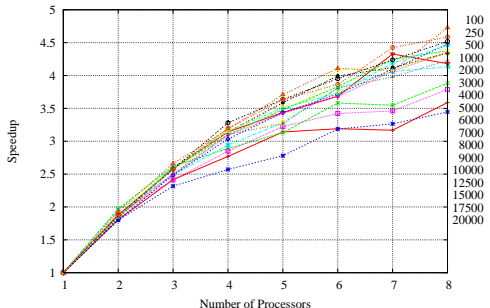
1.2: Then $m_{ik} = \frac{SUM[k]}{N}$ for $k = 1, 2, \dots, n$

Step 6. Calculate B^{-1}

I: Compute the Monte Carlo inverse $B^{-1} = MB_1^{-1}$

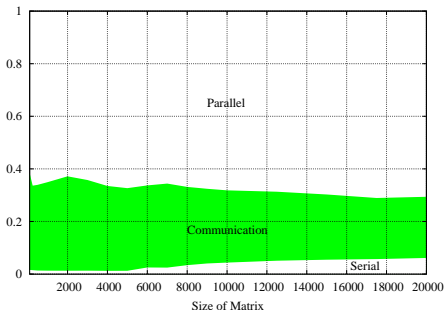
Parallel Approach

- Generation of Markov chains make up a big part of algorithm
- Chains can be generated independently
- This makes them an ideal candidate for parallelization
- Fairly good speedup achieved



Parallel Approach

- MPI used for parallel implementation
- Lower communication overhead for larger matrix sizes



- 1 Introduction to Monte Carlo Method
 - Monte Carlo Method
 - Markov Chains
 - Quasi Monte Carlo Method
 - Example I
 - Example II
- 2 Monte Carlo Methods for Linear Algebra
 - Problem to Solve
 - Algorithm
 - Parallel Approach
- 3 Data Splitting for Parallel Linear Algebra Monte Carlo
 - Introduction
 - Motivation
 - Data Splitting
 - Communication
 - Implementation
 - Results
- 4 Conclusion

Data Splitting

This work focuses on using data splitting to increase the applicability of parallel linear algebra Monte Carlo - to avoid duplicating the data which is sent to the slave processes.

Implementation of standard parallel Linear Algebra Monte Carlo methods:

- master pre-processes input data
- master divides number of chains/number of processes
- master sends all required data (full matrix A , P) to slaves
- each slave computes their chains
- master process gathers all solution matrices
- master averages solutions

Motivation

Memory requirements for matrix inversion for each slave:

- Iteration matrix (n^2)
- Probability matrix (n^2)
- Solution matrix (n^2)
- For increased speedup probability lookup matrix may be used (n^2)
- Some other variables

Total memory usage $> 3n^2$

Motivation cont.

Some computers and devices are equipped with low main memory.

IBM's Blue Gene has only 512MB per node and 256MB main memory per processor. But it has a fast interconnect network.

If we wish to invert a dense matrix ($\approx 3n^2$) with double precision:

$$n \text{ for } 256\text{MB} = \sqrt{256 \cdot 10^6 / (8 \cdot 3)} \approx 3200$$

$$n \text{ for } 512\text{MB} = \sqrt{512 \cdot 10^6 / (8 \cdot 3)} \approx 4600$$

Data Splitting Methods

Some possible ways of splitting matrices

Blocking

1
2
3
4

Round Robin

1
2
1
2

Random

1
2
2
1
1
2
1
2

Overlap

1
1
1
1 / 2
1 / 2
2
2
2

Communication

Different types of communication to exchange the state of chains are possible:

No communication no communication between the nodes.

Splitted chain communication node jumps within its part of data for a while and then sends state of chain to next node.

Blocked full communication node collects states of chains in blocks if they are not part of its rows.

Full communication node sends each individual state of chain to the corresponding node if next row is not part of its rows.

Data Splitting MC Method I

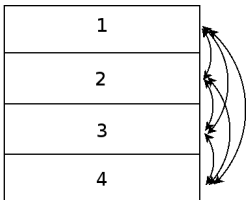
Blocked data splitting without communication

1
2
3
4

- Each node gets only part of rows of iteration matrix A and probability matrix P .
- Each node computes complete number of chains.
- Each node computes corresponding rows of solution matrix using only locally available data.

Data Splitting MC Methods II

Blocked data splitting with full communication

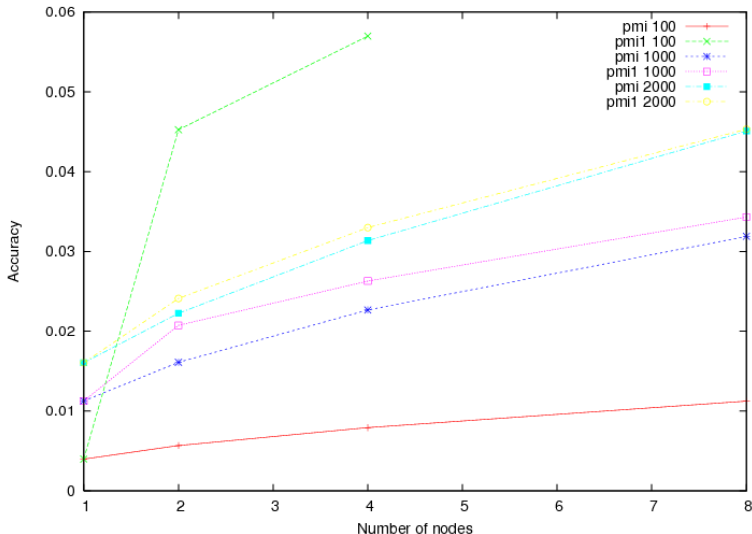


- Each node gets only part of rows of iteration matrix and probability matrix.
- Each node computes complete number of chains.
- Nodes send states of chains which want to jump to non local rows to corresponding nodes.

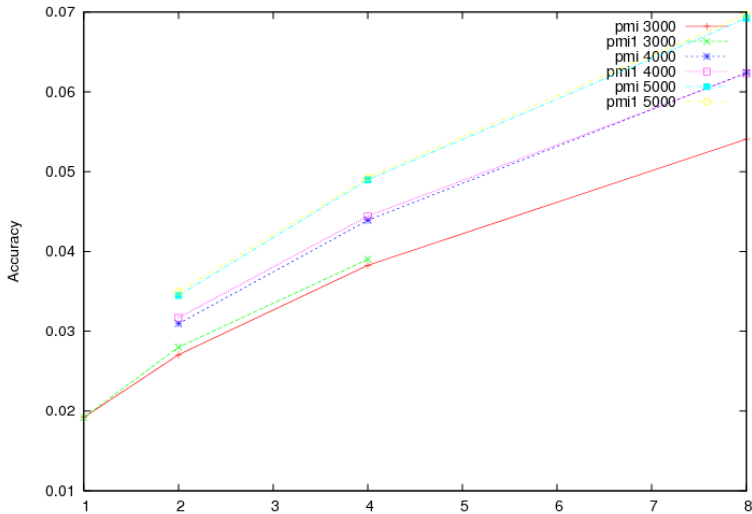
State of chain which has to be send consists of: starting row number, new row number, weight, partial solution vector.

Christian
Weihrach

Results I



Results II



Results Summary

- Blocked data splitting without communication implementation gives good results for diagonal dominant matrices which have larger dimension than 1000.
- Implementation with complete communication has following problems:
 - Communication between the nodes is asynchronous process.
 - Problem is that sends and receives have no pattern and are not predictable.
 - Deadlocks possible; could be avoided by use of non-blocking MPI functions
 - Difficulty is that the time the chain spends on each node may not be balanced.
 - **Decission:** No implementation with complete communication because it is too complex and would require too much communication.

- 1 Introduction to Monte Carlo Method
 - Monte Carlo Method
 - Markov Chains
 - Quasi Monte Carlo Method
 - Example I
 - Example II
- 2 Monte Carlo Methods for Linear Algebra
 - Problem to Solve
 - Algorithm
 - Parallel Approach
- 3 Data Splitting for Parallel Linear Algebra Monte Carlo
 - Introduction
 - Motivation
 - Data Splitting
 - Communication
 - Implementation
 - Results
- 4 Conclusion

Conclusion

- Monte Carlo Method can be used in many scientific applications
- Allows development of efficient parallel algorithms
- The combination of parallelization and Quasi MC allows development of well performing MI and SLAE solvers
- Blocked data splitting without communication implementation gives good results
- Implementation with complete communication is complex and requires too much communication.

Future Work

- Implementation of random and overlapping data splitting.
- Implementation of low communication methods.
- Increased communication effects have to be analysed.
- Look out for other solutions...

Thank You!

Questions...

`c.weihrauch@reading.ac.uk`

This work was supported by IPP- BAS, Grant BIS 21++ .