Reducing the Run-time of MCMC Programs by Multithreading on SMP Architectures

Jonathan M. R. Byrd    Stephen A. Jarvis    Abhir H. Bhalerao

Department of Computer Science
University of Warwick

MTAAP  IPDPS 2008
Outline

1. Markov Chain Monte Carlo
   - Introduction to MCMC
   - Program Cycle
   - Existing Parallelisation

2. Speculative Moves
   - Method
   - Theoretical Results
   - Practical Results
What is Markov Chain Monte Carlo?

- MCMC is a computationally expensive iterative technique for sampling from a probability distribution.
- Basic idea:
  - Construct a Markov Chain such that its stationary distribution is equal to the distribution we wish to sample.
  - After sufficient burn-in time, sampling from the chain is equivalent to sampling from the distribution.
What uses Markov Chain Monte Carlo?

- MCMC is widely used in
  - Bayesian statistics
  - Computational physics
  - Computational biology

- Specific applications include:
  - Phylogenetic analysis
  - Spectral modelling of X-ray data from the Chandra X-ray satellite
  - Calculating financial econometrics
  - Mapping vascular trees from retinal slides
The MCMC Program Cycle

1. **Let $x$ be the current state**
2. **Create a potential new state, $y$**
3. **Calculate acceptance probability $\alpha$**
4. **$\text{rng}() < \alpha$**
   - **No**
   - **Yes**
5. **Apply state change $x \rightarrow y$**
The MCMC Program Cycle

1. Let $x$ be the current state.
2. Create a potential new state, $y$.
3. Calculate acceptance probability $\alpha$.
4. If $\text{rng}() < \alpha$, then Yes; otherwise No.
5. If Yes, apply state change $x \rightarrow y$.
The MCMC Program Cycle

Let $x$ be the current state

Create a potential new state, $y$

Calculate acceptance probability $\alpha$

$rng() < \alpha$

Yes

Apply state change $x \rightarrow y$

No
The MCMC Program Cycle

- Let $x$ be the current state
- Create a potential new state, $y$
- Calculate acceptance probability $\alpha$
- If $\text{rng}() < \alpha$
  - Yes
  - Apply state change $x \rightarrow y$
  - No
The MCMC Program Cycle

Let \( x \) be the current state

Create a potential new state, \( y \)

Calculate acceptance probability \( \alpha \)

\( \text{rng}() < \alpha \)

Yes

Apply state change \( x \rightarrow y \)

No
The MCMC Program Cycle

Let $x$ be the current state

Create a potential new state, $y$

Calculate acceptance probability $\alpha$

$\text{rng}() < \alpha$

Yes

Apply state change $x \rightarrow y$

No
The MCMC Program Cycle

1. Let $x$ be the current state.
2. Create a potential new state, $y$.
3. Calculate acceptance probability $\alpha$.
4. If $\text{rng}() < \alpha$, Yes; if No, No.
5. If Yes, Apply state change $x \rightarrow y$.
Existing Parallelisation

- Execute multiple chains. Take samples from all of them.
  - Embarrassingly parallel
  - Does not reduce burn-in time.
  - Does not help escape local optima.
- Metropolis-Coupled MCMC
  - Execute multiple chains.
  - Coarse parallelisation, machines connected by LAN.
  - Modifies algorithm to improve rate of convergence.
  - Good for escaping local optima.
  - Hard to predict benefits.
Existing Parallelisation

- Execute multiple chains. Take samples from all of them.
  - Embarrassingly parallel
  - Does not reduce burn-in time.
  - Does not help escape local optima.

- Metropolis-Coupled MCMC
  - Execute multiple chains.
  - Coarse parallelisation, machines connected by LAN.
  - Modifies algorithm to improve rate of convergence.
  - Good for escaping local optima.
  - Hard to predict benefits.
Introducing Speculative Moves

1. Each state in a Markov Chain must depend on only the preceding state.
2. But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3. Consecutive rejected iterations could have been performed in parallel.
4. $\Rightarrow$ Assume all iterations will be rejected.
Introducing Speculative Moves

1 Each state in a Markov Chain must depend on only the preceding state.
2 But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3 Consecutive rejected iterations could have been performed in parallel.
4 $\Rightarrow$ Assume all iterations will be rejected.
Introducing Speculative Moves

1. Each state in a Markov Chain must depend on only the preceding state.
2. But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3. Consecutive rejected iterations could have been performed in parallel.
4. $\Rightarrow$ Assume all iterations will be rejected.
Introducing Speculative Moves

1. Each state in a Markov Chain must depend on only the preceding state.
2. But, typically only $\frac{1}{4}$ of iterations accept the proposed state-change.
3. Consecutive rejected iterations could have been performed in parallel.
4. $\Rightarrow$ Assume all iterations will be rejected.
Speculative Moves Program Cycle

1. Create a potential state $a$
2. Calculate acceptance probability $\alpha_a$
3. $\text{rng}() < \alpha_a$
   - Yes: Let $y = a$
   - No
4. Apply $x \rightarrow y$
5. Let $x$ be the current state
Speculative Moves Program Cycle

Thread A
Create a potential state $a$
Calculate acceptance probability $\alpha_a$

$rng() < \alpha_a$

Yes
Let $y = a$

No

Thread B
Create a potential state $b$
Calculate acceptance probability $\alpha_b$

$rng() < \alpha_b$

Yes
Let $y = b$

No

Apply $x \rightarrow y$

Let $x$ be the current state
Markov Chain Monte Carlo
Speculative Moves
Summary

Method
Theoretical Results
Practical Results

Speculative Moves Program Cycle

Thread A
Create a potential state \( a \)
Calculate acceptance probability \( \alpha_a \)
\( \text{rng}() < \alpha_a \)
Yes
Let \( y = a \)

Thread B
Create a potential state \( b \)
Calculate acceptance probability \( \alpha_b \)
\( \text{rng}() < \alpha_b \)
Yes
Let \( y = b \)

Thread C
Create a potential state \( c \)
Calculate acceptance probability \( \alpha_c \)
\( \text{rng}() < \alpha_c \)
Yes
Let \( y = c \)

Apply \( x \to y \)

Let \( x \) be the current state

Jonathan M. R. Byrd, Stephen A. Jarvis, Abhir H. Bhalerao
Multithreading MCMC
Theoretical Benefits of Speculative Moves

Let:
- \( n \) be the number of iterations considered concurrently.
- \( p_r \) be the average state-change rejection probability.

Each program cycle performs \( 1 \ldots n \) MCMC iterations.

On average \( \frac{1-p_r^n}{1-p_r} \) MCMC iterations are performed at each loop of the program cycle.

If multithreading overhead negligible, time for 1 program cycle \( \approx \) time for 1 MCMC iteration.
Theoretical Results

Maximum benefit of speculative moves on runtime

- 2 processes
- 4 processes
- 8 processes
- 16 processes

Runtime (percent of sequential)

$p_r$ (move rejection probability)
Theoretical Results

Maximum benefit of speculative moves on runtime

- 2 processes
- 4 processes
- 8 processes
- 16 processes

$p_r$ (move rejection probability) vs. RUNTIME (percent of sequential)
Practical Testing

- Circle detection algorithm used for testing
  - Fixed number of iterations
  - Autogenerated images
  - Runtime values averaged over 20 runs

- Hardware utilised:
  - AMD Athlon 64 X2 4400+ (dual-core)
  - Intel Xeon Dual-Processor
  - Intel Pentium-D (dual core)
  - Intel Core2 Quad Q6600 (2x dual-core dies)
  - 56 Itanium2 processor SGI Altix
Comparing Practical with Theoretical (1)

Runtime against $p_r$ on a Intel Pentium D (Dual-core)

- **Runtime (seconds)**
- **$p_r$ (move rejection probability)**
- 1 thread
- observed 2 threads
- theoretical 2 threads

Jonathan M. R. Byrd, Stephen A. Jarvis, Abhir H. Bhalerao

Multithreading MCMC
Comparing Practical with Theoretical (2)

Runtime against $p_r$ on an Intel Core2 Q6600 (Quad-core)

- 1 thread observed
- 2 threads observed
- 4 threads observed
- 2 threads theoretical
- 4 threads theoretical

$\text{Runtime (seconds)}$ vs. $p_r$ (move rejection probability)
Preferable Architectures

<table>
<thead>
<tr>
<th>System</th>
<th>2 threads</th>
<th>4 threads</th>
<th>8 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2 4400+ (2 core)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xeon (2 proc)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pentium D (2 core)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q6600 (4 core)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SGI Altix (56x Itanium2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Jonathan M. R. Byrd, Stephen A. Jarvis, Abhir H. Bhalerao
Multithreading MCMC
Will I Benefit?

This table shows the iteration time at which the overhead from multithreading balances the benefits, when $p_r = 0.75$.

<table>
<thead>
<tr>
<th>Processor Configuration</th>
<th>Iteration Time ($\mu s$)</th>
<th>Iteration Rate ($s^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xeon Dual-Processor</td>
<td>70</td>
<td>14 285</td>
</tr>
<tr>
<td>Pentium-D (dual core)</td>
<td>55</td>
<td>18 181</td>
</tr>
<tr>
<td>Q6600 (using 2 threads)</td>
<td>75</td>
<td>13 333</td>
</tr>
<tr>
<td>Q6600 (using 4 threads)</td>
<td>25</td>
<td>40 000</td>
</tr>
</tbody>
</table>
The speculative moves method uses increasingly available multiprocessor and multicore machines to reduce the runtime of MCMC program.

The statistical algorithm is preserved. Speculative moves will not effect the results, only the real-time required to obtain them.

Real-time reductions of 35% using a dual-core and 55% using quad-cores machines have been demonstrated.