Improving Operational Intensity in Data Bound Markov Chain Monte Carlo

Balazs Nemeth, Tom Haber, Thomas J. Ashby, and Wim Lamotte

EDM, Hasselt University, Belgium
Exascience Lab, imec, Belgium

Introduction

Background

- Processor stalls due to discrepancy between memory bandwidth and processor speed
- Efficiency determined by operational intensity: $\frac{\# \text{operations}}{\text{byte}}$
- Perform useful computation during otherwise stalled cycles
- Demonstrated with two Markov chain Monte Carlo samplers on Bayesian Logistic Regression target in big data and machine learning context

$$P(X|\theta) = \prod \frac{1}{1 + e^{x \theta}}$$

Memory Hierarchy

- Memory hierarchy places importance on locality
- Multiple cores needed to saturate memory bandwidth
- Per core memory bandwidth diminishes due to shared on-die memory controller
- Operational intensity tends to drop if $\#\text{cores}$ increases

Sampler Algorithms

Multiple Chain

- Evolve multiple independent chains in parallel
- Discard some fraction, $\rho$, of samples from each chain as burn-in
- Speedup from parallelism is limited by $1/\rho$

Multiple Proposal

- Evolve a single chain
- Sample from set of proposals around current proposal
- Discard $\rho$ fraction of samples from chain as burn-in

Original vs Restructured Algorithms

<table>
<thead>
<tr>
<th>Original algorithms</th>
<th>Restructured algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each core accesses all data</td>
<td>Data is partitioned across cores</td>
</tr>
<tr>
<td>Multiple Chain: Each core runs an independent chain ($#\text{chains} = #\text{cores}$)</td>
<td>Multiple Chain: Each core works on part of each chain ($#\text{chains} \neq #\text{cores}$)</td>
</tr>
<tr>
<td>Multiple Proposal: Each core evaluates the whole Bayesian likelihood at a proposal ($#\text{chains} = #\text{proposals}$)</td>
<td>Multiple Proposal: Each core evaluates part of the Bayesian likelihood at each proposal ($#\text{proposals} \neq #\text{cores}$)</td>
</tr>
<tr>
<td>Stalled cycles while waiting on data to move from memory to CPU caches</td>
<td>Perform useful work during otherwise stalled cycles</td>
</tr>
</tbody>
</table>

Results

Multiple Chain

- $\#\text{chains} = \#\text{cores}$ optimal

Multiple Proposal

- $\#\text{chains} = \#\text{proposals}$ optimal

Conclusion and Future Work

- Reducing $\#\text{chains}$ with the multiple chain sampler and more than twice as many proposal as cores with the multiple proposal sampler gives optimal performance
- Likelihoods that cannot be factored is part of future work
- Automatic tuning of $\#\text{chains}$ and $\#\text{proposals}$ will be studied
- Applying methodology to other algorithms will be explored next

Acknowledgments

Part of the work presented in this paper was funded by Johnson & Johnson. This project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 671555.