Computational Statistics and Data Analysis xx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

SIMD parallel MCMC sampling with applications for big-data Bayesian analytics☆

Q1 Alireza S. Mahani [a,*], Mansour T.A. Sharabiani [b]

[a] Scientific Computing Group, Sentrana Inc., USA

[b] National Heart and Lung Institute, Imperial College London, UK

ARTICLE INFO

Article history:
Received 11 October 2013
Received in revised form 9 February 2015
Accepted 12 February 2015
Available online xxxx

Keywords:
GPU
Hierarchical Bayesian
Intel Xeon Phi
Logistic regression
OpenMP
Vectorization

ABSTRACT

Computational intensity and sequential nature of estimation techniques for Bayesian methods in statistics and machine learning, combined with their increasing applications for big data analytics, necessitate both the identification of potential opportunities to parallelize techniques such as Monte Carlo Markov Chain (MCMC) sampling, and the development of general strategies for mapping such parallel algorithms to modern CPUs in order to elicit the performance up the compute-based and/or memory-based hardware limits. Two opportunities for Single-Instruction Multiple-Data (SIMD) parallelization of MCMC sampling for probabilistic graphical models are presented. In exchangeable models with many observations such as Bayesian Generalized Linear Models (GLMs), child-node contributions to the conditional posterior of each node can be calculated concurrently. In undirected graphs with discrete-value nodes, concurrent sampling of conditionally-independent nodes can be transformed into a SIMD form. High-performance libraries with multi-threading and vectorization capabilities can be readily applied to such SIMD opportunities to gain decent speedup, while a series of high-level source-code and runtime modifications provide further performance boost by reducing parallelization overhead and increasing data locality for Non-Uniform Memory Access architectures. For big-data Bayesian GLM graphs, the end-result is a routine for evaluating the conditional posterior and its gradient vector that is 5 times faster than a naive implementation using (built-in) multi-threaded Intel MKL BLAS, and reaches within the striking distance of the memory-bandwidth-induced hardware limit. Using multi-threading for cache-friendly, fine-grained parallelization can outperform coarse-grained alternatives which are often less cache-friendly, a likely scenario in modern predictive analytics workflow such as Hierarchical Bayesian GLM, variable selection, and ensemble regression and classification. The proposed optimization strategies improve the scaling of performance with number of cores and width of vector units (applicable to many-core SIMD processors such as Intel Xeon Phi and Graphic Processing Units), resulting in cost-effectiveness, energy efficiency ('green computing'), and higher speed on multi-core x86 processors.

1. Introduction

Many inference problems in statistics and machine learning are best expressed in the language of probabilistic graphical models, where a probability distribution function (PDF) over a high-dimensional parameter space can be motivated as the

☆ Source code accompanying the research presented in this paper, as well as links to software and hardware used/referenced in the paper, can be found in the online Supplementary material (see Appendix A).

* Correspondence to: Sentrana Inc., 1725 I Street NW Suite 900, Washington DC, 20006, USA. Tel.: +1 202 507 4524.
E-mail address: alireza.s.mahani@gmail.com (A.S. Mahani).

http://dx.doi.org/10.1016/j.csda.2015.02.010
0167-9473/

Please cite this article in press as: Mahani, A.S., Sharabiani, M.T.A., SIMD parallel MCMC sampling with applications for big-data Bayesian analytics. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.02.010

product of a collection of terms, each a function of a subset of the parameters. Inference in such models requires summarizing the joint PDF, which can be quite complex and lacking closed-form integrals. Monte Carlo Markov Chain (MCMC) sampling techniques offer a practical way to summarize complex PDFs for which exact sampling algorithms are not available. For many real-world problems, MCMC sampling can be very time-consuming due to a combination of large data sets, high dimensionality of joint PDF, lack of conjugacy between likelihood and prior functions, and poor mixing of the MCMC chain. Fast MCMC sampling is, therefore, important for wider adoption of probabilistic models in real-world applications.

For many years, software developers could rely on faster processors to see improved performance, without any need for code modification. In the past decade, however, chip manufacturers have warned of single-core performance saturation as CPU clock rates and instruction-level parallelism reach their physical limits (Jeffers and Reinders, 2013). Instead, we are seeing a steady rise in core counts and width of vector units, culminating in the emergence of many-core Single-Instruction-Multiple-Data (SIMD) architectures such as Graphic Processing Units (GPUs) and Intel Xeon Phi. Today, a CPU purchased for around $3000 can offer a theoretical peak double-precision performance of more than 300 GFLOPS, and many-core processors selling for nearly the same price have broken the Tera FLOP barrier. For scientific computing applications to take full advantage of such enormous performance potential within a single compute node, the entire parallelism potential of an algorithm must be efficiently exposed to the compiler, and eventually to the hardware.

There is a rich body of literature regarding parallel computing in general (see, e.g., Parkinson, 1987; Foster, 1995; Kirk and Wen-mei, 2012; McCool et al., 2012), and statistical and numerical analysis in particular (see, e.g., Kontoghiorghes, 2000, 2005). Most efforts on parallelizing MCMC algorithms, however, have focused on identifying high-level parallelism opportunities such as concurrent sampling of conditionally-independent nodes (Wilkinson, 2010). Such coarse-grained parallelism is often mapped to a distributed-memory cluster or to multiple cores on a shared-memory node. As such, vectorization capabilities of the processor are implicitly assumed to be the responsibility of libraries and compilers, resulting in a systemic under-utilization of vectorization in scientific computing applications. Furthermore, with increasing data sizes and widening gap between floating-point performance and memory bandwidth, modern processors have seen an architectural change in memory layout from Symmetric Multi-Processors (SMPs) to Non-Uniform Memory Access (NUMA) designs in order to better scale total memory bandwidth with the rising core count. The software-level, shared-memory view of this asymmetric memory is a convenient programming feature but can lead to a critical memory bandwidth under-utilization for data-intensive applications. This paper seeks to expand our ability to make efficient use of multicore x86 processors – today's most ubiquitous computing platform – to rapidly generate samples from the posterior distribution of model parameters. Our focus is two-fold: (1) identifying opportunities for SIMD parallelism in MCMC sampling of graphical models, and (2) efficiently mapping such SIMD opportunities to multi-threading and vectorization parallel modes on x86 processors. Using examples from directed and undirected graphs, we show that off-the-shelf, multi-threaded and vectorized high-performance libraries (along with vectorizing compilers) provide a decent speedup with small programming effort. Additionally, a series of high-level source-code and runtime modifications lead to significant additional speedup, even approaching hardware-induced performance limits. Vectorization of SIMD parallelism opportunities are often complementary with coarse-grained parallel modes and create a multiplicative performance boost. Moreover, we illustrate the counter-intuitive result that, given a limited number of cores available, efficient fine-grained (SIMD) multi-threading can outperform coarse-grained multi-threading over a range of data sizes where L3 cache utilization is the dominating factor. In the process, we propose a general strategy for optimally combining a sequence of maps to minimize multi-threading overhead while maximizing vectorization coverage. This strategy naturally allows for data locality at the memory and cache level, and significantly reduces the cost of complex synchronizations such as reduction on arrays. Furthermore, a differential update strategy for MCMC sampling of graphical models is proposed which, where applicable, can lead to significant reduction in data movement as well as the amount of computation, applicable to graphs with continuous as well as discrete nodes.

The remainder of this paper is organized as follows. In Section 2 we lay the theoretical foundation for the paper, including a summary of previous research on parallel MCMC, and an overview of two single-chain parallelization techniques for parallel MCMC of Directed Acyclic Graphs (DAGs). In Section 3 we do a detailed performance analysis and optimization of parallel MCMC for Bayesian GLM (focusing on logistic regression). In Section 4 we discuss several extensions to the ideas developed in Section 3, including Hierarchical Bayesian (HB) GLM, calculation of derivatives, Ising model, batch RNG, distributed and many-core computing, and compile-time loop unrolling. Section 5 contains a summary of our results and pointers for future research.

Given the extensive and specialized nature of performance and optimization analyses presented in the paper, we have included high-level summaries of lessons and implications for high-performance software development throughout the paper. These summaries are labelled as **Design Guideline**. A table containing a list of all design guidelines can be found in the Summary section (Fig. 19).

2. Parallel MCMC for graphical models

2.1. Previous work on parallel MCMC

A straightforward method for parallel MCMC is to run multiple chains (Wilkinson, 2010). Since each chain must go through the burn-in period individually, multi-chain parallelization is less suitable for complex models with poor

Please cite this article in press as: Mahani, A.S., Sharabiani, M.T.A., SIMD parallel MCMC sampling with applications for big-data Bayesian analytics. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.02.010