Electrical Power and Energy Systems 83 (2016) 33–39

Contents lists available at ScienceDirect

Electrical Power and Energy Systems

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

A novel GPU-accelerated strategy for contingency screening of static security analysis

Gan Zhou [a,*], Xu Zhang [a], Yansheng Lang [b], Rui Bo [c], Yupei Jia [b], Jinghuai Lin [d], Yanjun Feng [a]

[a] School of Electrical Engineering, Southeast University, Nanjing 210096, Jiangsu, China
[b] China Electric Power Research Institute, Beijing 100192, China
[c] Midcontinent Independent System Operator, Eagan, MN 5512, USA
[d] Fujian Electric Power Dispatching and Control Center, Fuzhou 350003, Fujian, China

ARTICLE INFO

Article history:
Received 13 April 2015
Received in revised form 28 March 2016
Accepted 29 March 2016
Available online 12 April 2016

Keywords:
Static security analysis
Contingency screening
GPU
Accelerated
Parallel computing
CUDA

ABSTRACT

Graphics processing unit (GPU) has been applied successfully in many computation and memory intensive realms due to its superior performances in float-pointing calculation, memory bandwidth and power consumption, and has great potential in power system applications. Contingency screening is a major time consuming part of contingency analysis. In the absence of relevant existing research, this paper is the first of its kind to propose a novel GPU-accelerated algorithm for direct current (DC) contingency screening. Adapting actively unique characteristics of GPU software and hardware, the proposed GPU algorithm is optimized from four aspects: data transmission, parallel task allocation, memory access, and CUDA (Compute Unified Device Architecture) stream. Case studies on a 3012-bus system and 8503-bus system have shown that the GPU-accelerated algorithm, in compared with its counterpart CPU implementation, can achieve about 20 and 50 times speedup respectively. This highly promising performance has demonstrated that carefully designed performance tuning in conjunction with GPU programing architecture is imperative for a GPU-accelerated algorithm. The presented performance tuning strategies can be applicable to other GPU applications in power systems.

Introduction

In recent years, graphics processing unit (GPU) has been used to accelerate the scientific calculation in many realms, such as petroleum survey, computational finance and computational fluid dynamics. Compared with same generation CPU, GPU has following significant advantages: Its floating-point calculation capability and memory bandwidth are about 10 and 5 times of CPU, respectively. Moreover, the power consumption of GPU is only about 20% of CPU when performing the same floating-point computation amount [1,2]. Therefore, how to apply GPU in power system has drawn more and more attentions [3].

In many power system computation, such as direct-current power flow (DCPF), alternating-current power flow (ACPF) and transient stability analysis, solving large-scale sparse linear system (SLS) is the most time-consuming part. For example, in ACPF analysis, over 80% of the time is spent on solving SLS and the rest is spent on generating Jacobian matrix [4]. In order to reduce the SLS solving time, Refs. [4–10] studied the GPU-accelerated strategies for common SLS solving algorithms, such as LU factorization, conjugate gradient (CG) iteration method, Jacobi iteration and multifrontal method. In those literatures, 3–10 times speedup has been reported. In addition, in order to improve the overall performance of ACPF analysis, Refs. [11,12] also studied the GPU-accelerated strategies for Jacobi matrix generating except SLS solving. In conclusion, GPU has been applied successfully in power system analysis and some of them have achieved significant performance improvements relative to their in-house CPU implementations. However, it is important to note that little work has been reported in existing literatures about how to design and optimize algorithms in conjunction with the unique hardware and software architecture of GPU, which is indeed the most critical factor to improve algorithm performance.

The N-1 static security analysis (SSA) is used to check steady-state security of the power grid when a single facility is in outage. For a system with N elements, the strict N-1 analysis requires to calculate N ACPFs, which implies huge computational cost for a large-scale power system. For example, the East China power grid owns about ten thousand nodes and the number of expected

* Corresponding author. Tel.: +86 136 0145 6838; fax: +86 25 8379 1696.
E-mail address: zhougan2002@seu.edu.cn (G. Zhou).

http://dx.doi.org/10.1016/j.ijepes.2016.03.048
0142-0615/

contingencies is more than 10,000. On traditional single-CPU computing platform, a complete N-1 analysis for such scale system will take more than 100 s, which does not meet the requirements of on-line analysis. In reality, there are only a few serious contingencies that will threaten the system security, so DCPF-based contingency screening is always carried out in a computationally efficient manner. There are mainly two methods for DCPF-based contingency screening. One method contains a ranking step in which all contingencies are sorted into a priority list by specific severity indices. As the subsequent ACPF analyses will be performed one by one in order of the priority list and can be quit anytime when satisfying terminal condition, it is very appropriate for single-CPU serial computing mode. However, the aforementioned quit operation maybe causes the missing of critical contingency, called masking phenomenon. In contrast, through analyzing specific severity indices, the other method without priority ranking generates a critical contingency set (CCS) whose size can be extended flexibly by adjusting index threshold. At the cost of increasing the CCS size and the corresponding calculation amount of ACPF, this method can effectively reduce the missing rate of contingency screening and is often applied in parallel computing mode, such as computer cluster and GPU [13–15]. This paper focuses on utilizing GPU to accelerate the latter contingency screening method.

Through extensive testing, it has been found that the time consumed on the contingency screening takes up to 30% of the total time of N-1 SSA, which means that the SSA time can be significantly reduced if contingency screening is effectively accelerated [15]. As the DCPF-based contingency screening mainly involves the dense vector operation, it is very suitable to be accelerated by GPU parallel computation. In the absence of relevant existing research, this paper is the first of its kind to propose a novel GPU-accelerated algorithm for DC contingency screening. In conjunction with unique characteristics of GPU software and hardware, the proposed GPU algorithm is optimized from four aspects: data transmission, parallel task allocation, memory access, and CUDA stream. Case studies on a 3012-bus system and 8503-bus system have shown that the GPU-accelerated algorithm, in compared with its counterpart CPU implementation, can achieve about 20 and 50 times speedup respectively. This highly promising performance has demonstrated that carefully designed performance tuning in conjunction with GPU programing architecture is imperative for a GPU-accelerated algorithm. The presented performance tuning strategies can be applicable to other GPU applications in power systems.

DCPF-based contingency screening

In order to reduce the calculation amount and improve the computational efficiency, the contingency screening is often carried out to identify the critical contingencies that may jeopardize the system security. This small subset of total contingencies will be analyzed in greater detail using ACPF method. As the contingency screening focus more on the branch thermal violation rather than the nodal voltage magnitude violation, the DCPF model, which is a linear model simplified from nonlinear ACPF, is typically employed [15].

Modeling and algorithm

The nodal active power injection and the branch active power flow are represented as (1) and (2), respectively.

$$P_i = V_i \sum_j V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) \tag{1}$$

$$P_{ij} = V_i V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) - G_{ij} V_i^2 \tag{2}$$

where the subscript i and j are node number; V represents node voltage magnitude; θ_{ij} is the voltage phase difference between node i and j; G_{ij} *and* B_{ij} are the real and imaginary part of $(i,j)th$ element of bus admittance matrix.

The DCPF model can be derived with the assumptions that: $V = 1.0$; $G_{ij} \ll B_{ij}$; $\theta_{ij} = 0$; all shunt admittances equal zero. Then, Eqs. (1) and (2) can be simplified as follows [16].

$$\boldsymbol{P} = \boldsymbol{B}_{n\times n}\boldsymbol{\theta} \quad \text{or} \quad \boldsymbol{\theta} = \boldsymbol{X}_{n\times n}\boldsymbol{P} \tag{3}$$

$$P_{ij} = \frac{\theta_i - \theta_j}{x_{ij}} \tag{4}$$

where $\boldsymbol{B}$ is the bus susceptance matrix (excluding the slack bus); the subscript $n \times n$ represents the dimension of matrix (n is the total number of buses excluding slack bus); $\boldsymbol{P}$ is a $n \times 1$ vector of nodal active power injection; $\boldsymbol{\theta}$ is the $n \times 1$ vector of nodal voltage angle; $\boldsymbol{X} = \boldsymbol{B}^{-1}$ is the bus reactance matrix, which is a dense matrix; x_{ij} represents the reactance of branch between node i and j.

When a contingency happens under pre-contingency status $\boldsymbol{\theta}_0 = \boldsymbol{X}_0\boldsymbol{P}_0$ (also called basic state), Eq. (3) can be expressed as:

$$\boldsymbol{\theta}_1 = \boldsymbol{X}_1\boldsymbol{P}_1 = (\boldsymbol{X}_0 + \Delta\boldsymbol{X})(\boldsymbol{P}_0 + \Delta\boldsymbol{P}) = \boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta} \tag{5}$$

and

$$\Delta\boldsymbol{\theta} = \Delta\boldsymbol{X}\boldsymbol{P}_0 + \boldsymbol{X}_0\Delta\boldsymbol{P} + \Delta\boldsymbol{X}\Delta\boldsymbol{P} \tag{6}$$

where the subscript 0 and 1 represent pre-contingency status and post-contingency status, respectively; for simplicity, assume $\boldsymbol{X}_0 = \boldsymbol{X}$ in this paper; the symbol Δ represents the incremental changes from pre-contingency status to post-contingency status.

If a branch is added between nodes i and j, the reactance increment matrix ΔX can be calculated with branch adding method as following:

$$\Delta\boldsymbol{X} = \frac{\boldsymbol{X}_0\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{X}_0}{X_{ii} + X_{jj} - 2X_{ij} + \Delta x_{ij}} = c\boldsymbol{X}_0\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{X}_0 \tag{7}$$

where

$$c = \frac{1}{X_{ii} + X_{jj} - 2X_{ij} + \Delta x_{ij}};$$

$\boldsymbol{M} = \boldsymbol{e}_i - \boldsymbol{e}_j$, e_i and e_j are standard basis; Δx_{ij} is the reactance of added branch (note that if a branch is taken out, the value of Δx_{ij} takes negative value).

Based on (4)–(7), the typical computational process for contingency screening can be described as follows:

(a) The bus reactance matrix $\boldsymbol{X}$, the inversion of bus susceptance matrix $\boldsymbol{B}$, is a dense matrix and its calculation is very time consuming, especially for a large scale system. As a common practice, there is no need to calculate $\boldsymbol{X}$ in every SSA cycle. Instead, if some incremental changes have happened to the basic grid, the new reactance matrix can be calculated quickly through branch adding method. Only when a relatively large topology change has happened, we need to perform the inversion calculation again. As the latter happens rarely, the computation time of $\boldsymbol{X}$ can be neglected and not be counted into the total time.
(b) Use (7) to calculate $\Delta\boldsymbol{X}$.
(c) Use (6) and (5) to calculate the nodal voltage angle $\Delta\boldsymbol{\theta}$ and $\boldsymbol{\theta}_1$, respectively.
(d) Use (4) to calculate the branch active power flow and check its overload status.

When a contingency leads to system separation, the denominator of c will become zero. Thereby, it is easy for above-mentioned