Connection Science

ISSN: 0954-0091 (Print) 1360-0494 (Online) Journal homepage: www.tandfonline.com/journals/ccos20

High-performance computing for static security assessment of large power systems

Venkateswara Rao Kagita, Sanjaya Kumar Panda, Ram Krishan, P. Deepak Reddy & Jabba Aswanth

To cite this article: Venkateswara Rao Kagita, Sanjaya Kumar Panda, Ram Krishan, P. Deepak Reddy & Jabba Aswanth (2023) High-performance computing for static security assessment of large power systems, Connection Science, 35:1, 2264537, DOI: 10.1080/09540091.2023.2264537

To link to this article: https://doi.org/10.1080/09540091.2023.2264537

Published online: 04 Oct 2023.

CONNECTION SCIENCE
2023, VOL. 35, NO. 1, 2264537
https://doi.org/10.1080/09540091.2023.2264537

High-performance computing for static security assessment of large power systems

Venkateswara Rao Kagita[a], Sanjaya Kumar Panda[a], Ram Krishan[b], P. Deepak Reddy[c] and Jabba Aswanth[a]

[a]Department of CSE, National Institute of Technology Warangal, Warangal, India; [b]Department of EE, National Institute of Technology Warangal, Warangal, India; [c]Department of EE, Indian Institute of Technology Kharagpur, Kharagpur, India

ABSTRACT

Contingency analysis (CA) is one of the essential tools for the optimal design and security assessment of a reliable power system. However, its computational requirements rise with the growth of distributed generations in the interconnected power system. As CA is a complex and computationally intensive problem, it requires a fast and accurate calculation to ensure the secure operation. Therefore, efficient mathematical modelling and parallel programming are key to efficient static security analysis. This paper proposes a parallel algorithm for static CA that uses both central processing units (CPUs) and graphical processing units (GPUs). To enhance the accuracy, AC load flow is used, and parallel computation of load flow is done simultaneously, with efficient screening and ranking of the critical contingencies. We perform extensive experiments to evaluate the efficacy of the proposed algorithm. As a result, we establish that the proposed parallel algorithm with high-performance computing (HPC) computing is much faster than the traditional algorithms. Furthermore, the HPC experiments were conducted using the national supercomputing facility, which demonstrates the proposed algorithm in the context of $N-1$ and $N-2$ static CA with immense power systems, such as the Indian northern regional power grid (NRPG) 246-bus and the polish 2383-bus networks.

ARTICLE HISTORY
Received 23 March 2023
Accepted 25 September 2023

KEYWORDS
Contingency analysis; high-performance computing; large power systems; $N-1$ contingency; $N-2$ contingency; security assessment

Nomenclature/Notation

BHU	Banaras Hindu university
CA	Contingency analysis
CPUs	Central processing units
DC	Direct current
FDLF	Fast decoupled load flow
GOSF	Generator outage sensitivity factors
GPUs	Graphical processing units
HPC	High-performance computing

CONTACT Sanjaya Kumar Panda sanjaya@nitw.ac.in Department of CSE, National Institute of Technology Warangal, Warangal, 506004 India

IIT	Indian Institute of Technology
LOSF	Line outage sensitivity factors
MPI	Message passing interface
NRLF	Newton-Raphson load flow
NRPG	Northern regional power grid
PARAM	PARAllel machine
PI	Performance index
PVM	Parallel virtual machine
SIMD	Single instruction multiple data
SISD	Single instruction single data
SSA	Static security assessment

1. Introduction

The power system has become vulnerable to disruptions due to fast network growth, rising energy consumption and incorporation of renewable energy resources (Gholami et al., 2020; Nayak et al., 2022a, 2022c). Subsequently, the power system's static security assessment (SSA) encounters numerous challenges in these scenarios. More specifically, the $N-1$ steady-state security of a power system CA is one of the most critical components of power system design, planning and real-time operation (Mitra et al., 2016; Salem et al., 2019). For instance, a transmission line or transformer contingency or outage can produce overloads in neighbouring lines, a rapid surge or a fall in node voltage, which may lead to cascaded contingency. Note that contingency is an abnormal condition in power systems, which puts the entire system, or a portion of it, under stress/blackout (Davis & Overbye, 2010; Ejebe & Wollenberg, 1979; Irisarri & Sasson, 1981; Mikolinnas & Wollenberg, 1981). The line outage is one of the major aspects in determining the impact of significant contingencies. As a result, providing possible preventative and corrective are the steps to resolve system breaches (Mishra & Khardenvis, 2012). Therefore, high-performance computations are needed for CA to guarantee the power system's security and reliability (Li et al., 2009; Li & Weng, 2009; Nayak et al., 2022b, 2021; Yang et al., 2005; Zhou et al., 2013). Generally, load flow solutions are considered to evaluate the impacts of the component outage(s) in CA. Hence, heavy computation is a critical issue in the context of CA.

Security assessment plays a significant role in the reliable power supply of wide-area power systems. The SSA helps to make appropriate control and operational decisions during the power system's element outage(s). However, performing SSA using the conventional approach (like sequential approach without HPC) is computationally challenging for larger power systems. Alternatively, SSA is essential to determine the state and ensure the power system's stable, reliable and steady operation. It performs repeated load flow analysis, which provides a steady state solution of the power system, such as voltage magnitude, angles, bus injection and power flows transmission lines (Burada et al., 2016). It is noteworthy to mention that load flow analysis is a non-linear and computationally intense tool in the power system. There are various methods for load flow analysis, namely Gauss-Seidel, Newton-Raphson load flow (NRLF), fast decoupled load flow (FDLF) and others (Grainger et al., 2015; Kothari & Nagrath, 2003; Patra, 2015). Among these methods, the NRLF is the most accurate but computationally intensive, whereas the FDLF is slightly less accurate,

with a significant reduction in computational burden. Therefore, the FDLF methods are often used for SSA using CA (Grainger et al., 2015; Kothari & Nagrath, 2003).

Contingency in a power system is an outage or loss of an element or a component or a combination of components in the power system, such as generators, transformers, or transmission lines. The outage could be due to the system's planned service maintenance or unplanned failure (Patra, 2015). The load flow analysis determines the system's state during each component outage. Moreover, it is generally a software application that tests all possible contingency cases for a particular system. On the other hand, CA observes and records limit violations in transmission lines' power flows or bus voltages for each outage. It is important for taking corrective actions to ensure the stable and reliable power operations. For a system with N number of elements, $N-1$ CA refers to analysing the occurrence of any single element outage (Mitra et al., 2016; Salem et al., 2019). Similarly, $N-k$ CA refers to a simultaneous outage of k different elements. Moreover, $N-1-1$ CA refers to the sequential outage of two system elements, where the second outage occurs after the necessary remedial/corrective action is taken for the first contingency (Mitra et al., 2016).

In $N-1-1$ contingency, the operator has prior knowledge about the critical $N-1$ outages and corresponding actions to mitigate the effects of that outage. However, there can be possibilities of another outage after corrective action, which is referred to as $N-1-1$ contingency. These sequential outages could impact the system, leading to severe blackouts. To perform $N-1-1$ CA, we simultaneously perform an outage of two elements and investigate the limit violations in the transmission lines. Generally, the number of possible two-element combinations in N-element system is $N\times$ ($N-1$). Therefore, performing $N-1-1$ CA is computationally intensive for large power systems. As a result, $N-1-1$ CA requires computationally an efficient SSA algorithm (Zhang et al., 2021).

Researchers have proposed various algorithms in the literature to address the discussed computation problem. Burada et al. (2016) have proposed a contingency screening and ranking method using NRLF. Davis and Overbye (2010) have used line outage sensitivity factors (LOSF) and generator outage sensitivity factors (GOSF) for faster computation. Chen and McCalley (2005) have investigated the network topology-based CA. Kumar and Reddy (2015) have employed exact and precise methods to improve efficiency in terms of computational time. These papers have focussed on efficiently ranking all possible contingencies during SSA without omitting any critical case contingency. Various performance indices are considered in the literature for $N-1$ CA to estimate outage severity using post contingent values and approximate the limit violations (Brandwajn & Lauby, 1989; Ejebe & Wollenberg, 1979; Irisarri & Sasson, 1981; Mikolinnas & Wollenberg, 1981; Zaborszky et al., 1980). However, none of these papers has considered HPC to solve the $N-1$ and $N-1-1$ CA. Currently, CA involves a heavy load flow computation which is very difficult to update every few minutes, even with parallel computation. The requirement for faster updates is crucial because power grids operate closer to the edge to fulfil the increasing energy demand. Generally, contingency cases are relatively independent, so CA is inherently a parallel process. Mathematically, there is a relatively straightforward parallelisation path, but the issue with parallelisation schemes remains due to the uneven computation time of individual cases.

There are several ways to reduce computational overhead. (1) Efficient mathematical modelling (2) Parallel computations using HPC by dividing a large task into a set of smaller

tasks. HPC employs parallel computers to solve scientific and engineering problems. These computers can range in various sizes, from a single high-performance workstation with many processing units to massive systems with hundreds or thousands of processors. Many significant works are initiated by considering power systems and HPC (i.e. distributed, grid, multicore and GPU computing) (Green et al., 2011; Ramesh, 1996). Here, HPC approaches enable faster analysis in various scenarios, even if a large set of variables exist.

This paper proposes parallel computation of $N-1$ and $N-2$ CAs and compares them with sequential computation of $N-1$ and $N-2$ CAs. Moreover, the paper aims to develop a fast and efficient HPC algorithm for large power systems' SSA without compromising accuracy. The proposed parallel algorithm with HPC ensures large power systems' security and helps in reliable operations. In the above-discussed scenarios and taking advantage of the independent nature of the contingencies, a strategy based on performance indices is proposed for screening and ranking the severe contingencies. SSA is a crucial process that ensures the reliable and stable operation of electric power systems by analysing their stability, voltage profiles and power flow patterns under various operating conditions. With the increasing size and complexity of modern power systems, HPC has become indispensable in enhancing the accuracy and efficiency of these assessments. We highlight the significant contributions of this paper as follows.

(1) We propose computationally efficient parallel algorithms for multicore HPC in a message-passing interface (MPI) environment to perform a faster calculation with acceptable accuracy in SSA. The algorithms are evaluated in the national supercomputing facility available at the Indian Institute of Technology (IIT) (Banaras Hindu University (BHU)), Varanasi.
(2) The proposed parallel algorithm enables the effective coordination among multicore CPU and GPU combination for faster computation.
(3) A composite performance index considering both line flow and bus voltage indices is proposed for efficient screening and ranking of the critical contingencies in large power system.
(4) To avoid the inversion of Jacobian matrix in FDLF, Conjugate Gradient (CG) method is implemented without compromisation of SSA accuracy.
(5) The proposed algorithms for $N-1$ and $N-2$ CA are evaluated in various standard test systems and Indian power systems, namely IEEE 14, IEEE 57, IEEE 118, NRPG 246-bus and polish 2383-bus.

The remaining parts of this paper are organised as follows. Section 1 describes the related work. Section 2 presents the mathematical formulation of the contingency screening and ranking problem. The key algorithmic ideas of parallel computation for SSA are described in Section 3. Section 4 shows the experimental results and discusses various test systems. Section 5 presents the conclusion and future scope.

2. Related work

This section emphasises the state-of-the-art algorithms for CA and establishes the need for an efficient algorithm. Performance index (PI) based SSA methods compute post-contingency to identify the severity of the outage (Mitra et al., 2016). The literature also

witnesses various contingency screening algorithms based on partial or approximate network solutions that reveal the voltage and/or power flow violation levels (Brandwajn & Lauby, 1989). However, due to approximation and high nonlinear impact, these algorithms may provide an unrealistic severity (Zou et al., 2022). With the recent advancement of computational resources, performing a complete alternating current power flow for large systems has become computationally feasible and reliable (Salem et al., 2019). This motivates us to develop parallel algorithms for a fast, reliable and accurate assessment of $N-1$ and $N-2$ CA using parallel computing infrastructure.

Balduino and Alves (2004) have used both MPI and parallel virtual machine (PVM) to perform the SSA on the Brazilian power grid. Note that this power grid serves millions of customers. Many paradigms, including synchronous and asynchronous master-slave topologies, are used to sample massive contingencies. A pervasive grid method is used to define user-friendly software architecture for gathering data from electrical networks. Subsequently, the data is processed to mimic potential scenarios in a real electrical network (Morante et al., 2006). Huang et al. (2009) have analysed the applicability of HPC for massive CA and focussed on load balancing. They have considered 3 lakh contingencies. Every contingency case is simply a power flow analysis. Each contingency changes its admittance matrix with an incremental change from the base case, given a solved base case. To manage case allocation and load balancing, one processor is designated as the master process and the other processors as the slave processes. Green et al. (2011) have used both static and dynamic load balancing strategies.

To formulate the real-time CA, both the DC load flow (DCLF) and AC load flow (ACLF) have been used on HPC platforms, Zhou et al. (2016) and Roberge et al. (2017), respectively. The DCLF is a linear model of non-linear power systems with large assumptions. Though DCLF is constitutionally efficient, the accuracy and capability of the solution are insufficient, e.g. the inability to check voltage limit violations. On the contrary, the non-linear ACLF is more accurate but constitutionally complex. In the literature, several sophisticated methods have been implemented on GPU to address alternating current power flow (ACPF)-based real-time CA (RTCA), such as the Newton-Raphson (NR) method and the Fast Decoupled (FD) method (Huang & Dinavahi, 2018). In such complex power system problems, researchers have used methods using the latest technologies: artificial intelligence, machine learning, deep learning and high-performance computing (Hailu et al., 2023) to resolve the constraints associated with existing power system solutions. In Qian et al. (2022), $N-1$ security assessment has been carried out using a deep learning algorithm, namely a deep convolutional neural network, where the impact of renewable energy resources is assessed. However, $N-1-1$ security assessment is not addressed due to computational burden. Moreover, they have not used high-performance computing to reduce the computational complexities. Hassan et al. (2022) have shown the impact of N - k contingencies leading to cascading failure. They have used a stacked denoising auto-encoder to extract the features of power systems and evaluated using the IEEE 118 bus test system, which may not be faster for higher bus systems like Polish and Indian power systems. However, high-performance computing makes it feasible to deal with such large power systems. Therefore, this paper attempts to solve the problem using high-performance computing because it can efficiently and accurately handle complex nonlinear problems.

A thorough CA procedure contains three steps, namely contingency selection, parallel CA and post-processing of CA. Pattery and Hassainar (2013) have established the $N-1$ CA

Table 1. Summary of the related work.

Article	Advantage	Disadvantage
Mitra et al. (2016)	Strategies for calculating performance indices to evaluate the $N-1-1$ contingency are given.	Faster computation method like HPC is not used.
Zhou et al. (2016)	HPC is implemented for DC load flow-based contingency analysis.	AC load flow-based security assessment is not considered.
Huang and Dinavahi (2018)	Data structure and kernel function are effectively used.	The parallel compensation method using a GPU cluster is not considered.
Zou et al. (2022)	An incidence matrix is proposed for the parallel calculation of $N-1$ contingency.	The large power system is not considered for evaluation.
Qian et al. (2022)	$N-1$ security assessment is only performed.	$N-1-1$ security assessment is not addressed. HPC is not used for implementation.
Hassan et al. (2022)	$N-k$ contingencies is only addressed.	The evaluation is limited to IEEE 118 bus test system. HPC is not used for implementation.
Hailu et al. (2023)	Advanced computational methods are reviewed.	Efficient system modelling is not focussed.

framework. They have explored and implemented computational load balancing strategies. Gopal et al. (2007) have explored the parallel implementation of direct current (DC) power flow-based CA on the GPU, which is quite faster than the CPU implementation. Ezhilarasi and Swarup (2009) have implemented a parallel processing methodology for faster calculations of SSA. However, its performance for a small system like the 14-bus system could be more efficient. The summary of the related work is shown in Table 1.

3. Contingency screening and ranking

Contingency ranking is based on the severity of the operational or element capacity limit violations. The severity calculation requires an intensive load flow solution after every single outage. Post-contingency load flow solution provides the bus voltage information and the power flow over the transmission line parameters (Gopal et al., 2007). These two parameters are crucial to know the system's operating condition. Given the line flow and voltage values at a particular contingency case, system constraints can be checked to ensure reliable operation using two performance indices (PIs), namely line flow index (PI_{pl}) and bus voltage index (PI_{vi}). In this work, the system condition is estimated by PI_c, which considers both PI_{pl} and PI_{vi} indices. The calculations of these indices are stated as follows (Prabhakar et al., 2022).

$$PI_{pl} = \frac{W_l^P}{2n}\left[\frac{P_l}{P_l^B}\right]^{2n} \tag{1}$$

$$PI_{vi} = \frac{W_i^V}{2n}\left[\frac{V_i - V_B^i}{\Delta V_i^{limit}}\right]^{2n} \tag{2}$$

$$PI_{Pl} + PL_{vi} = PI_c \tag{3}$$

where PI_{pl} is the PI of branch power flow, which indicates violations in l^{th} branch flow an outage, P_l is the power flow in l^{th} line during contingency, P_l^B is the base power flow value

for the l^{th} line, W_l^P is the weighting factor for l^{th} line, PI_{vi} is the voltage PI, which indicates voltage violations at i^{th} bus, V_i is the post contingency voltage at i^{th} bus, V_B^i is the base case voltage obtained in pre-contingency load flow at i^{th} bus, W_i^V is the weighting factor for i^{th} bus, ΔV_i^{limit} is the change in voltage and n is the empirical experimental co-efficient. If $PI_{pl} > 1$ or $PI_{vi} > 1$, then PI_c is incremented by 1.

The step-by-step procedure for ranking contingencies is as follows.

Step 1: Input line data and bus data of the test system.

Step 2: Perform FDLF analysis.

Step 3: Initialise all the possible contingencies in the network (say, $j = 1$).

Step 4: Run post-contingency load flow. Calculate PI_{pj} and PI_{vj} for every outage. If $PI_{pj} > 1$ or $PI_{vj} > 1$, then $PI_j = PI_j + 1$.

Step 5: Rank all the contingencies from the higher PI_j value to the least PI_j value.

It is noteworthy to mention that the load flow is a non-linear problem. However, its CA can be calculated by solving a set of linear equations $Ax = b$. In this paper, we use the FDLF method, which is faster with acceptable accuracy. Due to the decoupling of active and reactive power, the FDLF analysis is formulated with a constant Jacobian matrix and must not be calculated in each iteration. The basic FDLF model is discussed in the next section, and it is solved using a parallel programming approach.

4. Mathematical modelling and proposed algorithms using HPC

The convergence in the FDLF method is faster than the NRLF method and the Gauss-Seidel load flow method. It is often necessary to perform two to five iterations to achieve practical accuracy. The pace for iterations in the FDLF is nearly five times that of the NRLF method and roughly two-thirds that of the Gauss-Seidel method.

With the assumed slack bus voltage (usually, $V_1 = 1 \angle 0^\circ$ per unit), the remaining $(n-1)$ bus voltages are found through an iterative process. The process is described as follows (Prabhakar et al., 2022).

$$P_i = \sum_{j=1}^{n} |V_i||V_j||Y_{ij}| \cos(\theta_{ij} - \delta_i + \delta_j) \tag{4}$$

where P_i is the power injection at i^{th} bus, Y_{ij} is the admittance of line between i^{th} and j^{th} bus, θ_{ij} is the angle of Y_{ij} element of Ybus, δ_i is the voltage angle at i^{th} bus and δ_j is the voltage angle at j^{th} bus (Ahmadi et al., 2021).

$$Q_i = -\sum_{j=1}^{n} |V_i||V_j||Y_{ij}| \sin(\theta_{ij} - \delta_i + \delta_j) \tag{5}$$

where Q_i is the reactive power injection at i^{th} bus.

Equations (4) and (5) are called static load flow equations. The current injection I_i at i^{th} bus can be evaluated using active and reactive injected powers, which is defined as follows.

$$I_i = \frac{P_i - jQ_i}{V_i^*} \tag{6}$$

$$V_i = \frac{1}{Y_{ii}} \left(I_i - \sum_{j=1,\, j\neq 1}^{n} Y_{ij} V_j \right) \quad i = 1, 2, 3, \ldots, n \tag{7}$$

The voltage equation is formed with $(k+1)^{th}$ iteration (Ahmadi et al., 2021).

$$V_i^{(k+1)} = \frac{1}{Y_{ii}} \left[\frac{P_i - jQ_i}{(V_i^k)^*} - \sum_{j=1}^{i-1} \left(Y_{ij} V_j^{k+1} \right) - \sum_{j=i+1}^{n} \left(Y_{ij} V_j^k \right) \right] \tag{8}$$

The diagonal elements of Jacobian matrix is described as follows.

$$\frac{\partial P_i}{\partial \delta_i} = \sum_{j=1}^{n} |V_i||V_j||Y_{ij}| \sin(\theta_{ij} - \delta_i + \delta_j) - |V_i|^2 |Y_{ii}| \sin(\theta_{ii}) \tag{9}$$

By considering Equation (5), the above Equation (9) can be rewritten as follows.

$$\Rightarrow \frac{\partial P_i}{\partial \delta_i} = -Q_i - |V_i|^2 |Y_{ii}| \sin(\theta_{ii}) \tag{10}$$

$$\Rightarrow \frac{\partial P_i}{\partial \delta_i} = -Q_i - |V_i|^2 B_{ii} \tag{11}$$

where B_{ii} is the imaginary part of diagonal elements of Ybus.

As $B_{ii} >> Q_i$ and $|V_i|^2 \approx |V_{ii}|$, we can write

$$\Rightarrow \frac{\partial P_i}{\partial \delta_i} = -|V_i| B_{ii} \tag{12}$$

and

$$\frac{\partial P_i}{\partial \delta_i} = -|V_i| B_{ij} \tag{13}$$

where B_{ij} is the imaginary part of off-diagonal elements of Ybus.

$$\frac{\partial Q_i}{\partial |V_i|} = \sum_{j=1}^{n} |V_i||V_j||Y_{ij}| \sin(\theta_{ij} - \delta_i + \delta_j) - |V_i||Y_{ii}| \sin(\theta_{ii}) \tag{14}$$

As $B_{ii} = |Y_{ii}|\, sin(\theta_{ii})$, we can write

$$\frac{\partial Q_i}{\partial |V_i|} = Q_i - |V_i||Y_{ii}| \sin(\theta_{ii}) \tag{15}$$

$$\Rightarrow \frac{\partial Q_i}{\partial |V_i|} = Q_i - |V_i| B_{ii} \tag{16}$$

In compare to B_{ii}, Q_i is very less and can be neglected. Therefore, we can write

$$\frac{\partial Q_i}{\partial |V_i|} = |V_i| B_{ii} \tag{17}$$

Again, assuming $\theta_{ij} - \delta_i + \delta_j \approx \theta_{ij}$, we can write

$$\frac{\partial Q_i}{\partial |V_j|} = -|V_i| B_{ij} \tag{18}$$

and

$$\frac{\Delta P}{|V_i|} = -B' \Delta\delta, \frac{\Delta Q}{|V_i|} = -B'' \Delta|V| \tag{19}$$

where B' is the imaginary part of Ybus of order $(n-1) \times (n-1)$ and B'' is the imaginary part of Ybus of order $(n-1-n_{pv}) \times (n-1-n_{pv})$, where n_{pv} is the number of generator (PV) bus. Now, the value of $\Delta\delta$ and $\Delta|V|$ is obtained as follows (Ahmadi et al., 2021).

$$\Delta\delta = -[B'] \frac{\Delta P}{|V|} \tag{20}$$

$$\Delta|V| = -[B''] \frac{\Delta Q}{|V|} \tag{21}$$

We can solve voltage and delta from Equation (19), which are later used for computing the PI.

Our proposed algorithms use both CPU and GPU parallelisation and nodes like master, service and login. The rationality behind this is that it is based on three major factors. Firstly, MPI and CUDA are standardised, vendor-independent and portable libraries that are built for efficiency and flexibility. Secondly, it corresponds to the proposed algorithm implementation goal in real-world systems. Finally, it is simple to integrate with C language, making it ideal for code reuse, reduced development time and faster time to market.

4.1. Parallel computing with CPU and GPU

GPUs are single-instruction, multiple-data processors that have become a common characteristic of high-end video cards installed on general-purpose computers (Gopal et al., 2007). A computer strategy for attaining data level parallelisms (e.g. vector or array processor) is called single instruction multiple data (SIMD). As the name implies, a single instruction is applied to all data streams. On the other hand, there are significant differences between GPU and a CPU programming model. GPUs use SIMD processing, whereas the traditional CPU programming model uses single instruction single data (SISD) processing. Gopal et al. (2007) have investigated the parallel implementation of DC power flow-based CA on GPUs. They solved the power flow equations using Gauss-Jacobi iterations. Moreover, they have demonstrated that GPU implementation is significantly faster than CPU implementation. The GPU's heavily pipelined parallel architecture, as opposed to the CPU's serial architecture, is responsible for increasing the speed. Ezhilarasi and Swarup (2009) have presented a method for CA in power system security studies that use a parallel processing methodology. This methodology improves real-time experimentation by making it easier, faster and more accurate. A high-performance Linux cluster is used to run the experimentation. The testing was carried out using various IEEE standard test systems, namely the IEEE

14 bus, IEEE 30 bus, IEEE 118 bus, IEEE 162 bus and IEEE 300 bus. However, single-line interruptions are only taken into account for experimentation. Their method performs admirably in terms of speed and efficiency for large systems. However, they claim a performance reduction for very small systems, such as the IEEE 14 bus system. The algorithm's scalability is demonstrated through case studies. Although the number of iterations involved in the load flow process rises as the size of the system grows, this strategy still achieves a considerable parallel speedup.

4.2. Parallel implementation of N−1 static CA

Here, we discuss the implementation of the proposed parallel programming algorithm. The algorithm is shown in Algorithm 1, and the flow chart is depicted in Figure 1. We initialise the MPI environment, and every process is assigned a task. On the other hand, there is a process (called master) to coordinate all the tasks. Other processes are called slaves/workers. The processes are divided into two sets, master and slave, using MPI_COMM_WORLD. While the master is responsible for input and output, the slaves are responsible for their corresponding input and output in coordination with the master. The master process takes input bus data and line data from Ybus. On the contrary, the master process produces output as PI of contingencies. For this, the master performs pre-contingency load flow. From the pre-contingency load flow, we find the base voltage of buses, which is later used in the experimentation process of CA. Now, the master process sends the bus data, line data, Ybus and Base_V to all the slave processes for further experimentation process of CA. It is noteworthy to mention that rank is used to identify each processor, and size is used to determine the number of processors. Then we use an iterator *i_contg*, which is helpful to simulate all line outages and perform the load flow analysis.

Now we discuss how the work is divided between all the processes. Once the master process sends the required data to all the slave processes, the works are divided between the processes using their rank. Note that rank determines the id of the process. In general, a rank is an integer number, ranging from 0 to (size − 1), and the size is determined using MPI_Comm_size. A rank of a specific process can be determined using MPI_Comm_rank. All the processes iterate through *i_contg*. Here, we use an additional if statement to check whether a particular process should do the work. More specifically, we use a condition $if(i\_contg \,\%\, size \;==\; rank)$. This condition determines whether the process should enter inside and do the work. The line outage can be simulated by removing the line, performing the load flow analysis, calculating the PI and adding the removed line for the next experimentation of load flow analysis as in the sequential algorithm. However, in this algorithm, we see multiple contingencies simulating the load flow analysis of line outages in parallel at the CPU level of parallelism. Similarly, we perform the generator outage by changing the generator's real power and reactive power to zero, i.e. removing, analysing and adding the generator. We can calculate the PI of the generator outage after performing the load flow analysis. The master process receives PIs from each slave process. The above process is repeated for each slave process, and finally, the slave process sends PI to the master. We can also see from Figure 1 that we formulate B' and B'' that are being computed in GPU. Four kernels are used to form B' and B'' due to dependency between diagonal and off-diagonal elements. According to the Algorithm 1, we can see that the master process and

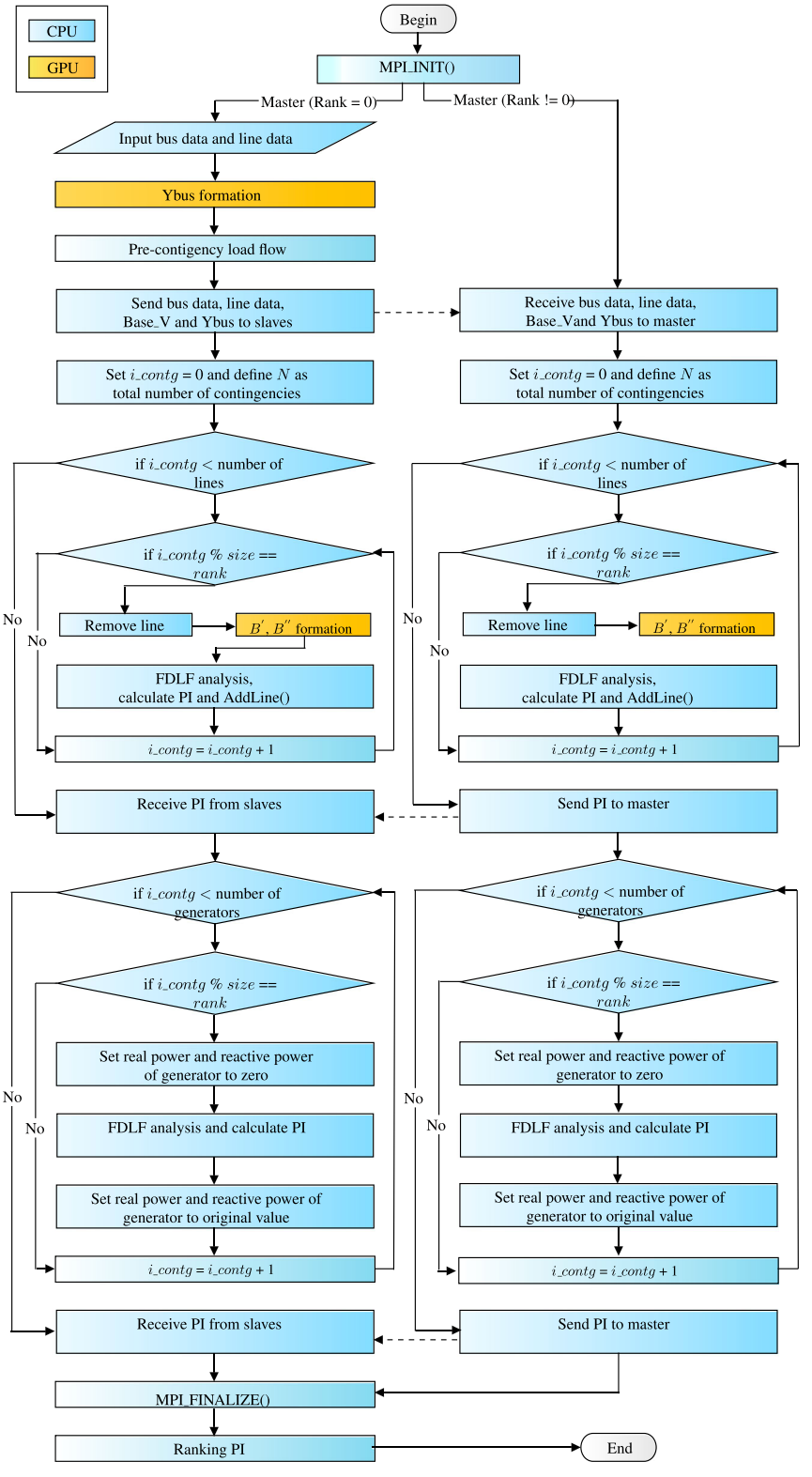

Figure 1. Parallel implementation flowchart for $N-1$ CA.

Algorithm 1: Parallel algorithm for $N-1$ static CA

```
Input  : Bus data and line data of power system
Output: PI of contingencies
Begin master processor
    // rank is used to identify each processor.
    // size determines number of processors.
    for each slave processor do
        Send bus data and line data
    end
    for each contingency i_contg do
        if i_contg % size == rank then
            RemoveLine()                          // Removes line i_contg
            Perform load flow analysis and find PI AddLine()      // Adds line
             i_contg
        end
    end
    for each contingency i_contg do
        if i_contg % size == rank then
            RemoveGenerator()              // Removes Generator i_contg
            Perform load flow analysis and find PI AddGenerator()      // Adds
             Generator i_contg
        end
    end
    for each slave processor do
        Receive PIs from slave
    end
Begin slave processor
    Receive bus data and line data
    for each contingency i_contg do
        if i_contg % size == rank then
            RemoveLine()                          // Removes line i_contg
            Perform load flow analysis and find PI AddLine()      // Adds line
             i_contg
        end
    end
    for each contingency i_contg do
        if i_contg % size == rank then
            RemoveGenerator()              // Removes generator i_contg
            Perform load flow analysis and find PI AddGenerator()      // Adds
             generator i_contg
        end
    end
    Send PI to master
```

slave processes are sharing the workload and processing them parallelly. Moreover, we can see the MPI implementation.

4.3. Parallel implementation of N−2 static CA

The proposed $N-2$ static CA algorithm is similar to the $N-1$ static CA and is shown in Algorithm 2. But, we have to simulate an outage of two lines (i.e. one line and one generator or two generators in the power system) in $N-2$ static CA. As we know, it is a very rare condition that two components fail simultaneously. Therefore, we first perform the $N-1$ CA by considering only the analysis's top five percent severe contingencies. Those lines are further considered for $N-2$ static CA. The rationality behind this is that we have a massive number of lines, and analysing all the lines is time-consuming. However, we consider all the generators for the $N-2$ static CA in the experimentation process. We can observe that we perform $N-1$ static CA (Algorithm 1) in the Algorithm 2 to find out the most severe cases. As we can see from the Algorithm 2, after checking the condition *if* ($i\_contg \,\%\, size == rank$) and removing one line, the process of $N-2$ contingencies is initiated. Note that it is represented in i_contg_1 and i_contg_2, respectively. Once the process is over, we add the removed line. In a similar fashion, we perform the generator outage by removing, analysing and adding the generator. Then we sort the PI of the contingencies to pick out the severe contingencies and only simulate the top five percent severe cases from the $N-1$ CA as part of the $N-2$ CA. It is noteworthy to mention that $N-2$ Contingency is a rare condition, but the possibility is high, mainly for the most severe cases from $N-1$ CA.

5. Experimental results

This section discusses the experiments of the $N-1$ and $N-2$ contingency, performs the analysis, and compares the efficacy of the proposed parallel algorithms to the performance of sequential algorithms. The experiments are performed on the PARAllel Machine (PARAM) Shivay supercomputing facility at the IIT (BHU), Varanasi. This facility contains Intel Xeon Skylake processors with NVIDIA Tesla V100. It also contains two master nodes, four service nodes, four login nodes and 223 CPU + GPU nodes and is assessed remotely. This facility is capable of producing 838 TFLOPS performance. We test the efficiency of the proposed algorithms for screening and ranking power system contingencies according to their expected severity on various standard test systems, such as IEEE 14 bus system, IEEE 57 bus system, IEEE 118 bus system, Polish 2383-bus system and NRPG 246-bus system (Kanpur, 2023). These IEEE test systems are publicly available at matpower.org. The possible contingencies range from 22 to 3025 for $N-1$ and 20 to 8335 for $N-2$, out of which it is important to identify the severe contingencies. Alternatively, these contingencies are ranked to show their severity. We used CUDA (version 11.4) and OpenMPI (version 4.1.2) to conduct the experimentations. We perform the experimentations using 20 MPI processes on a 10-core and 20-thread CPU. Table 2 ($N-1$ CA) and Table 3 ($N-2$ CA) present a comparative study between sequential and proposed parallel programming algorithms. The pictorial comparisons are shown in Figures 2 and 3 in which x-axis represents the dataset and y-axis represents the time in seconds. The dataset includes network parameters (line impedance and their connection topology), bus or node data (power injections), maximum and minimum limits of line capacity, active and reactive power limits of generation and constant

Algorithm 2: Parallel algorithm for $N-2$ static CA

```
Input  : Bus data and line data of power system
Output: PI of contingencies
// rank is used to identify each processor.
// size determines number of processors.
Begin master processor
    for each slave processor do
        Send bus data and line data
    end
    //Perform N − 1 analysis, rank contingencies from N − 1 analysis, and only use top five percent most severe contingencies of N − 1
      analysis for N − 2 analysis for each contingency i_contg do
        if i_contg % size == rank then
            RemoveLine()                                              // Removes line i_contg
            for each contingency i_contg_1 do                         // i_contg_1 > i_contg
                RemoveLine()                                          // Removes line i_contg_1
                Perform load flow analysis and find PI AddLine()      // Adds line i_contg_1
            end
            for each contingency i_contg_2 do                         // i_contg_2 > i_contg
                RemoveGenerator()                                     // Removes generator i_contg_2
                Perform load flow analysis and find PI AddGenerator() // Adds generator i_contg_2
            end
            AddLine()                                                 // Adds line i_contg
        end
    end
    for each contingency i_contg do
        if i_contg % size == rank then
            RemoveGenerator()                                         // Removes generator i_contg
            for each contingency i_contg_1 do                         // i_contg_1 > i_contg
                RemoveGenerator()                                     // Removes generator i_contg_1
                Perform load flow analysis and find PI AddGenerator() // Adds generator i_contg_1
            end
            AddGenerator()                                            // Adds generator i_contg
        end
    end
    for each slave processor do
        Receive PIs from slave
    end
end
Begin slave processor
    Receive bus data and line data for each contingency i_contg do
        if i_contg % size == rank then
            RemoveLine()                                              // Removes line i_contg
            for each contingency i_contg_1 do                         // i_contg_1 > i_contg
                RemoveLine()                                          // Removes line i_contg_1
                Perform load flow analysis and find PI AddLine()      // Adds line i_contg_1
            end
            for each contingency i_contg_2 do                         // i_contg_2 > i_contg
                RemoveGenerator()                                     // Removes generator i_contg_2
                Perform load flow analysis and find PI AddGenerator() // Adds generator i_contg_2
            end
            AddLine()                                                 // Adds line i_contg
        end
    end
    for each contingency i_contg do
        if i_contg % size == rank then
            RemoveGenerator()                                         // Removes generator i_contg
            for each contingency i_contg_1 do                         // i_contg_1 > i_contg
                RemoveGenerator()                                     // Removes generator i_contg_1
                Perform load flow analysis and find PI AddGenerator() // Adds generator i_contg_1
            end
            AddGenerator()                                            // Adds generator i_contg
        end
    end
    Send PI to master
end
```

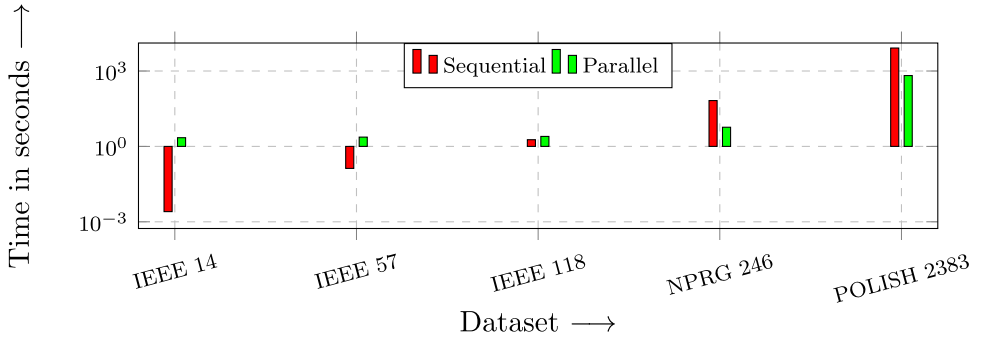

Figure 2. Logarithmic time comparison of $N-1$ sequential vs. parallel algorithm.

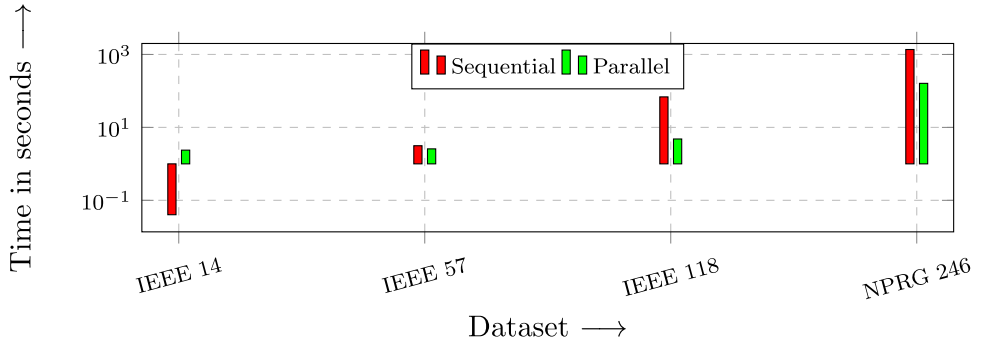

Figure 3. Logarithmic time comparison of $N-2$ sequential vs. parallel algorithm.

Table 2. $N-1$ static CA results.

$N-1$ Static CA				
		Time Elapsed (in seconds)		
Test Case	# of $N-1$ Contingencies	Sequential	Parallel	Speedup
IEEE 14	22	0.002538	2.218843	0.001144
IEEE 57	87	0.133219	2.343192	0.056854
IEEE 118	240	1.847536	2.498877	0.739347
NRPG-PGCIL 246	418	66.674126	5.791205	11.512997
Polish 2383	3025	8213.167969	659.667243	12.450471

loads. We show that the proposed algorithms outperform the sequential algorithm in terms of execution time for large power systems.

5.1. Test case: NRPG bus system

Indian NRPG system consists of five generators, 246 buses, 376 lines and 42 generating units. The number of $N-1$ contingencies and $N-2$ contingencies are 418 and 87,153, respectively. We test all the $N-1$ outages. The ranking of the top 10 contingencies is given in Table 4. The outages for which the Jacobian matrix is singular or unable to converge are considered as the most severe contingencies. Similarly, the top 10 critical $N-2$ contingencies in the NRPG system are listed in Table 5. Note that this paper aims to show the effectiveness

Table 3. $N-2$ static CA results.

$N-2$ Static CA				
		Time Elapsed (in seconds)		
Test Case	# of $N-2$ Contingencies	Sequential	Parallel	Speedup
IEEE 14	20	0.040291	2.37585	0.016959
IEEE 57	312	3.142429	2.576042	1.219867
IEEE 118	1845	68.636833	4.807695	14.276453
NRPG-PGCIL 246	8335	1366.535156	161.239768	8.475174

Table 4. Severe contingencies in Indian NRPG test System.

Contingency Type	ID	Flow Index (PI_{pl})	Voltage Index (PI_{vi})	System Index ($PI_{Pl} + PL_{vi} = PI_c$)	Severity Ranking
Line	140–144	925	1×10^{-5}	925	1
Generator	37	847	7×10^{-2}	847	2
Generator	20	632	6×10^{-2}	632	3
Line	140–143	608	3×10^{-5}	608	4
Line	139–145	550	5×10^{-5}	550	5
Line	139–152	404	2×10^{-4}	404	6
Line	54–55	394	1×10^{-3}	394	7
Line	121–122	381	6×10^{-2}	381	8
Generator	21	282	1×10^{-2}	282	9
Generator	19	273	2×10^{-2}	273	10

Table 5. Ranking of sever $N-2$ critical contingencies of Indian NRPG test system.

$N-2$ contingency Generator Outage (ID)	Line Outage (ID)	Flow Index	Voltage Index	System Index PI_{ci}	Rank
37	165–170	916	0.08	916.08	1
37	165–37	874	0.07	874.07	2
37	165–171	858	0.07	858.07	3
37	175–177	857	0.07	857.07	4
37	169–170	854	0.07	854.07	5
37	181–230	850	0.07	850.07	6
37	238–230	847	0.07	847.07	7
37	166–175	846	0.07	846.07	8
37	166–167	846	0.06	846.07	9
37	181–37	792	0.07	792.07	10

of the parallel algorithm over the sequential algorithm in power system applications. Both algorithms are tested using HPC clusters. Although other parallel algorithms can be used for power system applications, we have demonstrated only one such parallel algorithm. It can be further explored to select the most appropriate parallel algorithm in power system applications.

It is noteworthy to mention that we used five standard test systems, consisting of different complexities, such as topology, number of elements, nodes, thermal limit of transmission lines and many more. Moreover, these systems range from small to large power systems to show the elapsed time of the experiments. We observed the following impacts. (1) For the small test systems, the performance of the parallel algorithm is less effective than the sequential algorithm. (2) For the large test systems, the performance of the parallel

algorithm is noticeably better than the sequential algorithm. It is now clearly mentioned in Section 5.1.

6. Conclusion and future work

This paper has proposed a screening and ranking strategy of extensive contingencies in a faster mode of execution for the SSA of large power systems. A composite PI based on line flow and bus voltage performance under the contingency conditions has been implemented for contingency screening and ranking. FDLF with conjugate gradient linear equation solver has been implemented for the SSA of the large power system to achieve acceptable result accuracy with faster calculation. Although real-time CA is a significant energy management system component in many electric utilities, it has a huge computational overhead. We have proposed parallel algorithms for Multicore CPU and GPU-based HPC systems that simultaneously achieve both process-level and thread-level parallelism to cope with real-time requirements, i.e. fast calculation and accuracy. Accuracy and efficiency have been validated with numerous case studies on standard test systems, IEEE 14, 57, 118 and polish 2383 bus system, and field data of NRPG Indian 246 bus system. Further, calculation accuracy can be enhanced using the data volume for which an effective mathematical model may be designed in future work. Depending on the severity of the contingency, one can choose remedial action before considering a second outage. In summary, the parallel HPC-based CA is promising for industrial applications since it can simulate the whole $N-1$ and $N-2$ static CA for larger power systems like the Polish 2383 bus system (8335 possible contingencies) within a few minutes.

The proposed CA can play an essential role in the network design stages and in programmed maintenance or network expansion to detect network weaknesses. These weaknesses can be addressed by increasing transformer rating, transmission capacity and circuit breaker rating etc. Further, operator can prevent limit violations due to element outages by using appropriate remedial actions. The proposed strategy can be implemented for the $N-1-1$ contingency with GPU to achieve higher parallelism in our future work. On the other hand, the proposed algorithms are efficient only for larger power systems which may not be efficient for small power systems because the parallelisation process in the master processor and communication between GPU and CPU slaves take a considerably large time in comparison to calculation.

Acknowledgments

The authors would like to thank IIT (BHU), Varanasi, for providing the PARAM Shivay supercomputing facility to execute the HPC experimentations.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was fully supported by the National Supercomputing Mission (NSM), Department of Science and Technology (DST), Government of India (Reference No.: DST/NSM/R&D_HPC_Applications/2021/03.31).

References

Ahmadi, A., Smith, M. C., Collins, E. R., Dargahi, V., & Jin, S. (2021). Fast Newton-Raphson power flow analysis based on sparse techniques and parallel processing. *IEEE Transactions on Power Systems*, *37*(3), 1695–1705. https://doi.org/10.1109/TPWRS.2021.3116182

Balduino, L., & Alves, A. (2004). Parallel processing in a cluster of microcomputers with application in contingency analysis. In *IEEE/PES transmission and distribution conference and exposition: Latin America (IEEE Cat. No. 04EX956)* (pp. 285–290). IEEE.

Brandwajn, V., & Lauby, M. (1989). Complete bounding method for ac contingency screening. *IEEE Transactions on Power Systems*, *4*(2), 724–729. https://doi.org/10.1109/59.193806

Burada, S., Joshi, D., & Mistry, K. D. (2016). Contingency analysis of power system by using voltage and active power performance index. In *IEEE 1st international conference on power electronics, intelligent control and energy systems (ICPEICES)* (pp. 1–5). IEEE.

Chen, Q., & McCalley, J. D. (2005). Identifying high risk nk contingencies for online security assessment. *IEEE Transactions on Power Systems*, *20*(2), 823–834. https://doi.org/10.1109/TPWRS.2005.846065

Davis, C. M., & Overbye, T. J. (2010). Multiple element contingency screening. *IEEE Transactions on Power Systems*, *26*(3), 1294–1301. https://doi.org/10.1109/TPWRS.2010.2087366

Ejebe, G., & Wollenberg, B. (1979). Automatic contingency selection. *IEEE Transactions on Power Apparatus and Systems*, *1*, 97–109.

Ezhilarasi, G. A., & Swarup, K. (2009). Parallel contingency analysis in a high performance computing environment. In *International conference on power systems* (pp. 1–6). IEEE.

Gholami, M., Sanjari, M. J., Safari, M., Akbari, M., & Kamali, M. R. (2020). Static security assessment of power systems: A review. *International Transactions on Electrical Energy Systems*, *30*(9), e12432. https://doi.org/10.1002/etep.v30.9

Gopal, A., Niebur, D., & Venkatasubramanian, S. (2007). Dc power flow based contingency analysis using graphics processing units. In *IEEE Lausanne Power Tech* (pp. 731–736). IEEE.

Grainger, J., Stevenson, W., & Stevenson, W. (2015). Power systems analysis, 2nd education.

Green, R. C., Wang, L., & Alam, M. (2011). High performance computing for electric power systems: Applications and trends. In *IEEE power and energy society general meeting* (pp. 1–8). IEEE.

Hailu, E. A., Nyakoe, G. N., & Muriithi, C. M. (2023). Techniques of power system static security assessment and improvement: A literature survey. *Heliyon*, *9*(3), 1–18.

Hassan, R., Sun, R., & Liu, Y. (2022). Online static security assessment for cascading failure using stacked de-noising auto-encoder. *International Journal of Electrical Power & Energy Systems*, *137*, 107852. https://doi.org/10.1016/j.ijepes.2021.107852

Huang, S., & Dinavahi, V. (2018). Real-time contingency analysis on massively parallel architectures with compensation method. *IEEE Access*, *6*, 44519–44530. https://doi.org/10.1109/Access.6287639

Huang, Z., Chen, Y., & Nieplocha, J. (2009). Massive contingency analysis with high performance computing. In *IEEE power & energy society general meeting* (pp. 1–8). IEEE.

Irisarri, G., & Sasson, A. (1981). An automatic contingency selection method for on-line security analysis. *IEEE Transactions on Power Apparatus and Systems*, *PAS-100*(4), 1838–1844. https://doi.org/10.1109/TPAS.1981.316524

Kanpur, I. (2023). Northern regional power grid (NRPG) data. https://www.iitk.ac.in/eeold/facilities/Research_labs/Power_System/NRPG-DATA.pdf. Online; accessed 28 February 2023.

Kothari, D. P., & Nagrath, I. (2003). *Modern power system analysis*. Tata McGraw-Hill Publishing Company.

Kumar, U., & Reddy, H. (2015). Contingency ranking in modern power system by exact and precise method. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering*, *3*(5), 229–237.

Li, K.-C., Hsu, C.-H., Wen, C.-H., Wang, H.-H., & Yang, C.-T. (2009). A dynamic and scalable performance monitoring toolkit for cluster and grid environments. *International Journal of High Performance Computing and Networking*, *6*(2), 91–99. https://doi.org/10.1504/IJHPCN.2009.027459

Li, K.-C., & Weng, T.-H. (2009). Performance-based parallel application toolkit for high-performance clusters. *The Journal of Supercomputing*, *48*, 43–65. https://doi.org/10.1007/s11227-008-0204-2

Mikolinnas, T., & Wollenberg, B. (1981). An advanced contingency selection algorithm. *IEEE Transactions on Power Apparatus and Systems*, *PAS-100*(2), 608–617. https://doi.org/10.1109/TPAS.1981.316917

Mishra, V. J., & Khardenvis, M. D. (2012). Contingency analysis of power system. In *IEEE Students' conference on electrical, electronics and computer science* (pp. 1–4). IEEE.

Mitra, P., Vittal, V., Keel, B., & Mistry, J. (2016). A systematic approach to n-1-1 analysis for power system security assessment. *IEEE Power and Energy Technology Systems Journal*, *3*(2), 71–80.

Morante, Q., Ranaldo, N., Vaccaro, A., & Zimeo, E. (2006). Pervasive grid for large-scale power systems contingency analysis. *IEEE Transactions on Industrial Informatics*, *2*(3), 165–175. https://doi.org/10.1109/TII.2006.877266

Nayak, S. K., Panda, S. K., & Das, S. (2022a). Constrained-based power management algorithm for green cloud computing. *International Journal of Computational Science and Engineering*, *25*(6), 657–667. https://doi.org/10.1504/IJCSE.2022.127187

Nayak, S. K., Panda, S. K., & Das, S. (2022b). Unconstrained power management algorithm for green cloud computing. In *Advances in distributed computing and machine learning: Proceedings of ICADCML* (pp. 3–14). Springer.

Nayak, S. K., Panda, S. K., Das, S., & Pande, S. K. (2021). A renewable energy-based task consolidation algorithm for cloud computing. In *Control applications in modern power system: Select proceedings of EPREC* (pp. 453–463). Springer.

Nayak, S. K., Panda, S. K., Das, S., & Pande, S. K. (2022c). A multi-objective renewable energy-based algorithm for geographically distributed datacentres. *International Journal of Embedded Systems*, *15*(2), 119–131. https://doi.org/10.1504/IJES.2022.123304

Patra, K. K. (2015). Contingency analysis in power system using load flow solution. *International Journal of Computer Applications*, *975*(2), 8887.

Pattery, J., & Hassainar, S. (2013). High performance computing for contingency analysis of power systems. *International Journal of Engineering Research and Technology*, *2*(9), 1993–1998.

Prabhakar, P. S. V., Krishan, R., & Pullaguram, D. R. (2022). Static security assessment of large power systems under n-1-1 contingency. In *22nd National power systems conference (NPSC)* (pp. 35–40).

Qian, T., Shi, F., Wang, K., Yang, S., Geng, J., Li, Y., & Wu, Q. (2022). N-1 static security assessment method for power grids with high penetration rate of renewable energy generation. *Electric Power Systems Research*, *211*, 108200. https://doi.org/10.1016/j.epsr.2022.108200

Ramesh, V. (1996). On distributed computing for on-line power system applications. *International Journal of Electrical Power & Energy Systems*, *18*(8), 527–533. https://doi.org/10.1016/0142-0615(96)00016-6

Roberge, V., Tarbouchi, M., & Okou, F. (2017). Parallel power flow on graphics processing units for concurrent evaluation of many networks. *IEEE Transactions on Smart Grid*, *8*(4), 1639–1648. https://doi.org/10.1109/TSG.2015.2496298

Salem, F. K. A., Jaber, M., Abdallah, C., Mehio, O., & Najem, S. (2019). A distributed spatiotemporal contingency analysis for the lebanese power grid. *IEEE Transactions on Computational Social Systems*, *6*(1), 162–175. https://doi.org/10.1109/TCSS.2018.2888689

Yang, C.-T., Shih, P.-C., & Li, K.-C. (2005). A high-performance computational resource broker for grid computing environments. In *19th International conference on advanced information networking and applications (AINA'05) Volume 1 (AINA papers)* (Vol. 2, pp. 333–336). IEEE.

Zaborszky, J., Whang, K.-W., & Prasad, K. (1980). Fast contingency evaluation using concentric relaxation. *IEEE Transactions on Power Apparatus and Systems*, *PAS-99*(1), 28–36. https://doi.org/10.1109/TPAS.1980.319605

Zhang, Y., Guo, Q., Zhou, Y., & Sun, H. (2021). Online frequency security assessment based on analytical model considering limiting modules. *CSEE Journal of Power and Energy Systems*, *8*(5), 1363–1372.

Zhou, G., Zhang, X., Lang, Y., Bo, R., Jia, Y., Lin, J., & Feng, Y. (2016). A novel GPU-accelerated strategy for contingency screening of static security analysis. *International Journal of Electrical Power & Energy Systems*, *83*, 33–39. https://doi.org/10.1016/j.ijepes.2016.03.048

Zhou, R., Ai, Z., Yang, J., Chen, Y., Li, J., Zhou, Q., & Li, K.-C. (2013). A hypervisor for MIPS-based architecture processors-a case study in Loongson processors. In *IEEE 10th International conference on high performance computing and communications & 2013 IEEE international conference on embedded and ubiquitous computing* (pp. 865–872). IEEE.

Zou, Q., Luo, F., & Zhang, T. (2022). An incidence matrix based analytical method of n-1 contingency parallel analysis of main transformers in distribution networks. *CSEE Journal of Power and Energy Systems*.