Parallel computing in railway research
Qing Wu, Maksym Spiryagin, Colin Cole and Tim McSweeney
Centre for Railway Engineering, Central Queensland University, Rockhampton, Australia

ABSTRACT
Available computing power for researchers has been increasing exponentially over the last decade. Parallel computing is possibly the best way to harness computing power provided by multiple computing units. This paper reviews parallel computing applications in railway research as well as the enabling techniques used for the purpose. Nine enabling techniques were reviewed and Message Passing Interface, Domain Decomposition and Hadoop & Apache are the top three most widely used enabling techniques. Seven major application topics were reviewed and iterative optimisations, continuous dynamics and data & signal analysis are the most widely reported applications. The reasons why these applications are suitable for parallel computing were discussed as well as the suitability of various enabling techniques for different applications. Computing time speed-ups that were reported from these applications were summarised. The challenges for applying parallel computing for railway research are discussed.

1. Introduction
Computer simulations are a fundamental part of modern research. The development of many research disciplines is accompanied and, in most cases, facilitated by the ever increasing availability of computing power. This increasing availability is further contributed to by two trends, namely the ever increasing computer performance and the ever decreasing computer prices. Figure 1 shows the computing performance of the best (Top 1) and the 500 best (Top 500) supercomputers from 1993 to 2018 [1].

It can be clearly seen that their computing performance has been continuously increasing in an order of magnitude every four or five years. Data reported from [2] also show that the performance of individual computer processors has also been being improved since 1970s. Regarding computer prices, Reference [3] reported the manufacturing price indexes for imported, exported and locally traded computer equipment in the United States (US) market. This report [3] indicates computer prices have been decreasing not just in US but also globally.

Meanwhile, commercial cloud computing platforms, such as Amazon Web Services [4] and Google Cloud [5], are also contributing to an increase of the accessibility of supercomputing to the public. For researchers or institutions who do not have access to
a conventional cluster, cloud computing provides a viable solution. This paper will show that cloud computing has been very well adopted in railway data processing and some other railway research.

Having significant or super computing power available, the question naturally arises as to how we can harness the power and make the most of it. And there seems no better answer than parallel computing [6]. The concept of parallel computing is straightforward, using multiple computing units to process multiple tasks simultaneously. The literature review of this paper has found significant ambiguities in naming the basic computing units; various names have been used in different publications: computer, node, Central Processing Unit (CPU), core, process, thread. Actually, the items in that list have a progressively inclusive relationship. In other words, an item in the list can have multiple numbers of the items that come after it in the list. As all items could be used as basic parallel computing units, the name of ‘computing unit’ is used in this paper instead of any name that is restricted to a specific hardware or software level. A computing unit can be any item in the name list.

The primary benefits of parallel computing are the time saving and the possibility of better results. For example, due to the high computing efficiency, simulation models can be developed with more details for finite element analysis (FEA) [7] and more search iterations can be used for optimisation studies [8], with the outcome that better simulation results and optimisation results can be achieved.

The above discussions generally apply to all research disciplines including railway research. This paper focuses on railway research and reviews the applications of parallel computing in the field. The review gives an insight of the application status of parallel computing in railway research; it also helps to identify the trends and future of parallel computing in this field. Section 2 will introduce some basics of parallel computing while Section 3 introduces the enabling techniques that can be used to conduct parallel computing. Section 4 reviews the railway applications that have been found in open literature and discusses why these applications are suitable for parallel computing. Section 5 analyses the speed-ups that have been reported from the railway applications. Section 6 discusses the challenges in applying parallel computing for railway research.
The objective of this paper is to include all enabling techniques and applications that are directly related to railway related parallel computing; it has also included as many directly related publications as possible. However, it should be noted that an exhaustive review of all publications is never possible.

2. Parallel computing basics

The simplified concept of parallel computing and a comparison with the serial computing concept are shown in Figure 2. It can be seen that the essence of the parallel computing concept is using multiple computing units to simultaneously process multiple computing tasks. Note that the computing units do not need to be computer cores or CPUs, they can be any processing units that can conduct operations independently. For example, in references [9,10] the parallelised computing units are Arithmetic Logic Units (ALUs) which are components of a single computing core. Therefore, to be able to conduct parallel computing, the requirement for hardware is multiple computing units, in most cases, multiple computing cores. They can be made available from multi-core Personal Computers (PCs) or a network of computers (namely the clusters). Almost all modern PCs have more than one computing core, which means all modern PCs are capable of some level of parallel computing. Some institutions have clusters (also named supercomputers or High Performance Computing systems) available for their members and associates; clusters can also be set up by using multiple normal PCs linked via the Ethernet or accessed via commercial cloud computing platforms.

For a computing question to be processed using parallel computing, the question itself should be able to be partitioned into multiple independent computing tasks. The independence of the computing tasks can be of different levels. For example, when parallel computing is used to process matrix operations, as in Reference [11], communications among different computing tasks are required in each time-step. However, when parallel computing is used in parallel optimisations, as in Reference [12], communications are only required in each optimisation iteration. The level of independence of the computing tasks in the former case is lower than in the latter case. One of the most important factors to consider during computing task partitioning is the load balance [6] among different computing units. An ideal scenario is that the computing tasks are partitioned evenly so all computing units can finish their tasks at the same time so as to minimise the waiting.

Figure 2. Parallel and serial computing concepts.
time and consequently the total computing time. When load balance is not well con-
trolled, parallel computing can result in even longer computing time than serial comput-
ing as discussed in Reference [13].

Having partitioned the computing tasks, techniques are needed to coordinate and communicate the parallel computing process. Two different programming models are commonly used to facilitate the communication and coordination for the parallelised computing tasks, namely distributed memory and shared memory [6] as shown in Figure 3. In the distributed memory models, such as the Message Passing Interface (MPI) technique, each computing unit has its own individual memory, whilst in the shared memory models, such as the Open Multi-Processing (OpenMP) and POSIX threads (pthreads) techniques, all computing units usually share the same memory locations. Commonly, in both computing models, the coordination task is assigned to a specific computing unit (core, thread or process); the coordinating unit is commonly called the master unit while all other parallelised units are called the slave units. More parallel computing related techniques will be discussed in Section 3. Also, many other references, such as References [6] and [14], can be used for more general and detailed information about parallel computing.

3. Enabling techniques

This section reviews the parallel computing enabling techniques that were reported from railway related publications. Figure 4 shows the reported techniques with the corresponding number of publications that are reviewed in this section. Note that the publication numbers in Figure 4 do not cover all publications that are reviewed in this paper as, in some publications, the enabling techniques cannot be clearly identified. Further, the techniques shown in the figure do not exclude each other; one publication can use multiple enabling techniques. For example, the Domain Decomposition Method (DDM) technique has to be used with another communication technique such as the MPI to successfully conduct parallel computing. As shown in Figure 4, nine enabling techniques will be discussed in this section and MPI, DDM and Hadoop

![Figure 3. Parallel computing models: (a) distributed memory, and (b) shared memory.](image-url)
Apache are the top three most widely used enabling techniques. The Hadoop & Apache technique is mostly related to cloud computing.

Note that a special caution should be taken here for the analysis of the cloud computing technique. Cloud computing (mostly related to Big Data) is an extremely popular topic in modern research, and this statement also applies to railway research. There are a significant number of railway related publications for cloud computing, and cloud computing itself is inherently parallel computing related. However, most of the publications did not directly focus on parallel computing applications nor parallel computing techniques. The criterion used in this paper to select parallel computing related publications from cloud computing related publications is that a selected cloud computing related publication should have at least one occurrence of parallel computing in the text. Using this criterion, only ten publications are found that are specifically parallel computing focused (shown in Figure 4 as the Hadoop and Spark group). To discuss railway related cloud computing specifically, another paper is needed. Also, due to publication print limits, the techniques are only briefly discussed in this paper; more detailed information can be found in the corresponding references.

3.1. Domain Decomposition Method (DDM)

As mentioned in Section 2, for a computing question to be processed using parallel computing, the question itself should be able to be partitioned into multiple independent computing tasks. For computing questions that have naturally independent computing tasks, the partitioning process is straightforward. Some examples of such computing questions can be: (1) vector and matrix operations [15]; (2) traffic data processing [16]; (3) individual fitness evaluations in optimisations [12]; and (4) signal processing for multiple channels [17]. However, when the parallelised computing tasks are not independent, most commonly in FEA [7], Computational Fluid Dynamics (CFD) [18] and Discrete Element Analysis (DEA) [19], special techniques are needed to partition the computing question into parallelisable computing tasks. In this regard, the DDM [20] is possibly the most important technique. For example, Figure 5(a)
shows an array of independent computing tasks and each box represents one computing task. As there is no interconnection among the boxes, the partitioning of computing tasks can be easily done. However, in Figure 5(b) all boxes are linked to adjacent boxes, and boundary conditions (interfaces or domain communications) must be considered when partitioning the computing tasks for parallel computing. The DDM in this case can deal with the boundary conditions and partition the dependent computing tasks into parallelisable domains as shown in Figure 5(b). DDM has been widely used in railway applications [7,19,21–29]. More information regarding the DDM can be found in Reference [20].

3.2. Communication protocols

Having partitioned the computing tasks, communications are needed between slave computing units and the master unit, and also sometimes among the slave units. Many techniques, such as MPI and OpenMP, provide communication libraries or programming extensions to achieve the communications. In some studies, researchers also use communications protocols, i.e. Transmission Control Protocol/Internet Protocol (TCP/IP) [30–32] and Hypertext Transfer Protocol (HTTP) [33] to achieve the communication. However, it has to be pointed out that the TCP/IP and HTTP are only used for communication, and some other techniques have to be used to launch the parallelised computing tasks. For example, in Reference [32], TCP/IP was used for communication while OpenMP was used to launch the parallelised computing tasks.

3.3. MPI

MPI is a message passing standard that is used to facilitate parallel computing with the distributed memory model as shown in Figure 3(a). As can be seen from the distributed memory model in that figure, each computing unit has its own memory and the computing units are have a high level of independence from each other. Therefore, one of the most important functions of the MPI is to provide communications for the computing units. More information regarding MPI can be accessed via the online book of Barney [34]. In railway research, the MPI has found many applications [7,11–13,21–23,31,35–43]; it is one
of the most widely used parallel computing enabling techniques. In the authors’ opinion, MPI is very suitable for parallel co-simulations and parallel optimisations, as presented in References [31] and [42] respectively, since they have very independent parallelised computing tasks.

3.4. OpenMP

OpenMP is a programming extension that is designed primarily for parallel computing with the shared memory model as shown in Figure 3(b). Compared with the distributed model, the shared memory model used by OpenMP makes it more flexible when initialising and finalising parallel processes. A good illustration of the flexibility is shown in Figure 6 where, for example, one computing process contains a number of parallelisable matrix operations and parallelisable for-loops. When using OpenMP, as the memory is shared among all computing units, the computing program can easily fork into multiple parallelised processes or threads and then operate on the shared memory (shared data). Upon finalisation of the parallelised processes, all forked processes can then easily rejoin the master process. For railway research, OpenMP has found applications in [11,19,32,44,45]. More information regarding OpenMP itself can be found in Barney’s other online book Barney [46].

3.5. Multithreading

To better comprehend multithreading, one should understand the differences between a process and a thread. A simple explanation is that a thread is a part of a process and a process can have multiple threads. Parallel computing can be conducted among a number of processes as well as a number of threads. MPI and OpenMP can deal with both thread and process levels of parallel computing. In this section, the term ‘multithreading’ indicates only thread level parallel computing. There are a number of techniques that can be used to manipulate threads; two fundamental ones are Pthreads [47] in a Unix Operation System (OS) and the Windows Application Programming Interface (WinAPI) threads [48] in Windows OS. As multithreading deals with multiple threads that are in the same process, it understandably uses the shared memory parallel computing model as shown in Figure 3(b). Applications using Pthreads have been reported from [49,50]; and Reference [15] used WinAPI threads. Besides Pthreads and WinAPI threads, JAVA and Matlab also provide high level multithreading API. Several applications [8,51–53] using JAVA have been

![Figure 6. Illustration of the forking process.](image-url)
reported along with a few applications using Matlab [54–56]. Understandably, higher level APIs would be easier to use which can help to cut down the development cost. According to Barney [47], the Pthreads technique has the advantage of light weight which means less overhead cost (computing time in this case) in parallel computing coordination and communications. However, one inconvenient point about Pthreads is that it was developed in C/C++ programming language. For applications in other programming languages, such as FORTRAN, an additional interface must be developed.

3.6. **General-purpose computing on Graphics Processing Units (GPGPU)**

The GPGPU technique uses Graphics Processing Units (GPUs) for general purpose computing as conventional CPUs. GPUs generally have slower computing speed, but they have the advantage of a large number of computing units. For example, most modern CPUs accommodate a maximum eight cores each while a GPU nowadays can easily accommodate hundreds of computing cores. The Compute Unified Device Architecture (CUDA) parallel computing platform can enable GPUs to be used as GPGPUs; a few such railway applications [17,57,58] have been reported. Deriving from the original purpose of GPUs, the GPGPU technique is very suitable for graphic processing such as image processing in machine vision condition monitoring [17,57] and results analysis for FEA [58].

3.7. **Field-programmable gate array (FPGA)**

A FPGA is a reconfigurable and programmable digital chip that can be set up to control multiple computing units (CPUs, GPUs or others) in parallel. It provides an array of uniform Configurable Logic Blocks and Input/Output Blocks that can be programmed to form different circuits as designers want. It offers significant configuration flexibility and a large number of gate resources which contribute to its great capability of parallel computing. For example, researchers can programme eight of exactly the same circuits (on the same chip) that are connected to eight computing cores. Parallel computing can then be conducted by sending the computing cores parallelised computing tasks. Also, different circuits can be programmed and then connected to different computing cores, so that the FPGA and the computing cores can then be used to compute different computing tasks but still in a parallel fashion. Applications of parallel computing using FPGAs have been reported [59–62]. Most commonly, FPGAs are used to achieve real-time signal processing or real-time hardware-in-loop applications.

3.8. **Hadoop and Spark**

Commercial cloud computing platforms have provided another approach for parallel computing. In railway research, most cloud computing applications [16,63–67] have used the Hadoop MapReduce technique. The Hadoop framework has a file distribution system called Hadoop Distributed File System (HDFS) which can be regarded as a computing task partitioning tool from a general parallel computing perspective. Then another Hadoop technique called MapReduce can coordinate parallel computing for the distributed files on different computing nodes. One important feature of
Hadoop MapReduce is that it brings data processing algorithms to the parallelised nodes where the distributed data stored. This feature can significantly reduce the processing time when compared with moving data to the algorithms as data are significantly larger than algorithms. More information regarding MapReduce will be discussed in Section 4.3. Various applications [68–70] have used the MapReduce technique, but on non-commercial clusters.

In addition to Hadoop MapReduce, another cloud computing technique called Apache Spark has also found railway applications [71–73]. Within the Apache Spark framework, there are four main components: a driver node, a number of worker nodes, a cluster manager, and executors (the same number as for worker nodes). The driver node is basically a user interface which also does the computing task partitioning. The worker nodes (slave computing units) conduct the specific computing tasks in parallel. The cluster manager that is under the control of the driver node coordinates the parallel computing process while the executors manage the computing tasks within each of the worker nodes. Hadoop MapReduce and Apache Spark have different ways of moving data and are therefore most suitable for different applications. Hadoop MapReduce moves data via networks and disks, has slower computing speed and is most suitable for analysis of archived Big Data. Apache Spark caches data in memory, has faster computing speed and is most suitable for real-time data processing. Comparatively, Hadoop MapReduce would be very suitable to analyse data that are already collected from field tests, while Spark is better for processing of on-line condition monitoring data.

3.9. Others

Besides the previously reviewed techniques, there are some others that are also available and were used for railway research. Bosso and Zampieri [74]) used the LabVIEW software package to enable parallel computing while Desjouis et al. [75] and Diedrichs [18] used two other software packages called PETSc and STAR-CD respectively. Two special and very successful parallel computing applications were reported from References [9,10] where the researchers used the streaming extension of a Pentium processor to conduct Single Instruction Multiple Data parallel computing. The special part of these two applications is that they only used one computing core with parallelised ALUs. It has to be noted that the ALUs can only process simple numerical operations and the applications may have been outdated once multicore PCs were widely available.

4. Applications in railway research

This section discuss the applications of parallel computing reported in railway research and why these applications are suitable for parallel computing. Figure 7 shows the applications found in the literature and the corresponding number of publications. The reviewed publications are presented in seven major topics; iterative optimisations, FEA/CFD/DEA and data and signal analysis are the most widely reported applications. Again, special caution should be taken for the application of Big Data analysis (cloud computing related) as discussed in the beginning of Section 3. The publications allocated to the different topics shown in Figure 7 are not exclusive. Therefore, there
are publications that are related to multiple topics and are therefore allocated against multiple topics. It should also be noted that categorisation is always a challenging task, and the authors do not claim that Figure 7 is the best categorising approach; however, it is presented according to the authors’ best knowledge.

4.1. Iterative optimisation

Iterative optimisations, such as Particle Swarm Optimisations (PSO), Genetic Algorithm (GA) and Simulated Annealing (SA), improve a group of solutions (plans, designs, etc.) iteratively in loops. Almost all iterative optimisation methods have algorithm structures that are very suitable for parallel computing, especially during the fitness assessment step as shown in Figure 8. For each iteration of the optimisation process, the group of solutions, i.e. the individuals in the figure, have to go through the same assessment procedures to be assessed regarding their fitness (quality and performance of the solution). The assessments of individual solutions are completely independent and are therefore very suitable for parallel computing. Another two advantages of parallelising the fitness assessment step of iterative optimisations are: (1) the fitness assessment step is usually the most time consuming step and parallelising this step can help to minimise the serial parts of the algorithm.

Figure 7. Parallel computing applications in railway research.

Figure 8. Concept of parallel iterative optimisations [38].
so as to achieve high computing speed; and (2) Fitness assessments for each individual usually have exactly the same procedures and therefore have mostly the same computing loads, and this can help to archive great load balance among slave computing units which also helps to achieve high computing speed. Note that the second advantage does not apply to all optimisation cases. For example, when an optimisation of vehicle suspension parameters is considered, some suspension parameters can cause vehicle instability that requires longer computing time.

In railway research, parallel computing applications have been widely reported for train scheduling optimisations with different optimisation algorithms. Huang et al. [76] have used GA, Boman and Gerdovci [8] and Bettinelli et al. [43] used the Tabu optimisation method. Boman and Gerdovci [8] also used SA. Iqbal et al. [51, 52] used a combination optimisation algorithm. In addition to these applications, parallel GA has been used to optimise railway vehicle aerodynamics [77], truss structures [58], shock absorber distribution for railway tunnels [36], draft gear designs [37–39] and wedge suspension designs [42]. Parallel PSO has been used for optimisations of passenger traffic prediction [69], draft gear designs [37–39] and wedge suspension designs [42]. Evolutionary Algorithm optimisation has been used for optimisations of railway vehicle structures [33], and railway vehicle dynamics design [12]. A special parallel computing related optimisation study was conducted by Matsuura and Miyatke [56] to optimise train speed profile to achieve lower energy consumption, but they used parallel computing for dynamic programming instead of fitness assessment.

4.2. FEA/CFD/DEA

FEA and CFD commonly solve continuous problems by using a significant number of interconnected discrete nodes. For example, a square steel plate as shown in Figure 5(b) can be approximated in FEA by an array of nodes that are interconnected by springs and dampers. After this assumption, each node with mass property can be regarded as rigid and the force distribution within the steel plate can then be approximated by the force distribution among all the springs and dampers. In CFD, a fluid field can also be expressed by an array of nodes that are also similar to the illustration in Figure 5(b). The concept of DEA is an already discretised version of FEA or CFD. Example applications of FEA, CFD and DEA are railway bridge structural analysis [55], crosswind analysis [18] and ballast simulation [19] respectively. As explained in Section 3.1, FEA/CFD/DEA studies are also very suitable for parallel computing as: (1) they generally require simulating a large amount of elements and the interconnections among the elements are of high nonlinearity which creates computationally expensive simulation tasks; and (2) in many cases, the elements are homogeneous or very similar to each other, hence parallel computing can be successfully applied using the very well developed DDM. In railway research, parallel FEA has mainly been used to study three topics: (1) wheel-rail contact issues [7, 28, 29]; (2) bridge structures [25, 55]; and (3) interactions between trains and infrastructure [21–23, 26]. In addition to the FEA related studies, Hoang et al. [19, 24] have used parallel DEA to simulate railway ballast. Parallel CFD has been used to analyse vehicle response subject to cross-wind [18], the entry flow issue of railway tunnels [27], catenary aerodynamics [78] and air brake modelling [41].
4.3. Data and signal processing

Data processing, especially Big Data processing, is naturally suitable for parallel computing as: (1) Big Data require a lot computational resources; and (2) a Big Data set can be easily partitioned into multiple sub data sets. Most Big Data applications are also related to cloud computing as reviewed in Section 3.8 and the MapReduce technique is most commonly used. A good illustration of the MapReduce technique can be found in Reference [79] and the process is reproduced in Figure 9. A Big Data set is first partitioned and distributed to multiple computing nodes by using the HDFS technique. A mapper [79] is responsible to record the locations of the sub data sets and the relations among them. Then the reducers [79] will process the data in parallel, extract useful information and summarise it for output. In railway research, Hadoop MapReduce has been used to process passenger flow data so as to predict passenger traffic [67, 69]. It has also been used to process high speed train noise data [68], signal fault diagnosis data [65], railway equipment management data [66], urban railway general management data [16] and ground penetrating radar data [70].

In the topic of data processing, besides the utilisation of Hadoop MapReduce, the Apache Spark platform was used by Sangat et al. [71] to process geospatial data. Xie et al. [53] used a multithreading technique to process freight data from different regions in parallel where one computing unit was specifically used for the data from one region.

For signal and image processing, parallel GPUs and FPGA are often used. Parallel GPUs were used by Santur et al. [57] and Wang et al. [17] to process machine vision rail and catenary inspections respectively. In the former application, two GPUs were used to process the image captured by two cameras in parallel. In the latter application, three GPUs were used for image acquisition, processing and saving in parallel. Zhang [59] used FPGA to enable multiple computing units to process rail non-destructive testing data in parallel while Khan et al. [60] used FPGA to enable multiple computing units to process signals that were received from multiple sensors. In Reference [80], the researchers used seven computers to process the signals from 14 sensors that measured rail straightness.

4.4. Cloud computing

Cloud computing is closely related to Big Data processing that has been reviewed in Section 4.3. Most publications [16, 65–71] found in this topic have already been
reviewed in Section 4.3. Apart from these Big Data applications, two other applications [63,64] were found for railway power supply simulations. These two applications have also used the Hadoop MapReduce technique and more information about railway power supply simulations will be provided in Section 4.5.

4.5. Power network simulations

A number of railway (infrastructure) power network simulation studies [9,10,30,35,49,63,64,75,81] have been found that have also used parallel computing. Railway power network simulations use mathematical modelling to simulate the load flows and energy consumption of the railway power network. A common objective of these studies is to determine the supply capability of the power network or to determine the energy consumptions of certain railway operations. The key process of power network simulation is shown in Figure 10. Before the simulation, details of the infrastructure layout (power stations, transformers, etc.) are already known, hence circuit models can be developed using the infrastructure layout. During the simulation, the power supply process is discretised into a certain number of time instants or time-steps. For each time-step, i.e. different time instants, the locations and status of power consumers (the trains and their pantographs) can be searched from a database or input from other simulations. Having the network circuit model and the information regarding the trains, the power that is needed for train operations can be determined. As described, train locations and status at each time-step can be stored in a database and directly used during the simulation. This feature has made the simulation process ‘time independent’. In other words, the simulation of a certain time-step does not require information for the time-step before nor after. Therefore, power network simulations are also suitable for parallel computing. After the discretisation (by different time instants) of the simulation process, different computing units can be used to determine the power demand of different time periods. Then information from different computing units can be gathered to draw the final conclusions. The process is also suitable for the use of the Hadoop MapReduce technique as applied in References [63,64].

4.6. VSD

Comparatively, parallel computing applications in VSD simulations have more diverse parallelised computing steps as VSD modelling usually deals with more detailed components. Wheel-rail contact simulations are well recognised as one of the most time consuming steps for VSD simulations. Parallelising the simulation of wheel-rail contact becomes a natural idea when discussing parallel computing for VSD. The literature
review has found two approaches to conducting parallel computing for wheel-rail contact simulations. The first approach was used in References [74,82] where the researchers used one computing unit for each wheel-rail interface. The second approach by Vollebregt [44] took advantage of the forking process as shown in Figure 6 to increase the computing speed for numerical operations during wheel-rail contact simulations. The latest version of CONTACT [83] available to the authors has the function of using one computing unit for the computation of each contact point. Note that multiple contact points can exist for one wheel-rail interface. The forking process was also used by Kogut and Ciurej [11] and Pogorelov et al. [15] to simulate vehicle-track-soil interaction dynamics and locomotive-train interaction dynamics respectively.

In Reference [54], multiple computing units were used to solve kinematics equations in parallel, using one computing unit for the equation of each Degree of Freedom. In Reference [84], slab track, vibration analyses were conducted using parallel computing where different computing units were used to analyse the vibration results of different frequency bands.

The authors have also conducted a number of parallel computing to process VSD simulations. In Reference [13], parallel computing was used to determine various force elements in Longitudinal Train Dynamics simulations in parallel. In Reference [40], a three-dimensional long train system dynamics simulation was conducted where all vehicles in the train were modelled with wheel-rail contact and one computing unit was used to compute the dynamics of each vehicle. In Reference [32], three computing units were used to simulate three detailed locomotive models (in parallel) that then provided simulated traction forces for Longitudinal Train Dynamics simulations. In Reference [31], vehicle-track dynamics were studied and two computing units were used to simulate the vehicle and the track in parallel. Bosomworth et al. [50] also used parallel computing to conduct three-dimensional train dynamics simulations where the thermal effects of interaction between wheels and rails were studied.

### 4.7. Scheduling

Most parallel computing related scheduling studies are also related to iterative optimisation that has been reviewed in Section 4.1. Various studies [8,43,51,52,76] have used some optimisation algorithms. A study reported in Reference [85] has used the Depth-First Search (DFS) tree method to develop train scheduling plans. The DFS tree method is shown in Figure 11 in which the numbered nodes are scheduling tasks that have to be done and the branches that connect the nodes have different time and space properties. The goal of the DFS tree method is to find the routes or scheduling plans that have the shortest travel distances or shortest travel times. Some branches are not related with each other at all, e.g. the three branches that connect nodes 2, 7 and 8 to node 1. Three searchers can be dispatched from node 1 to search the tree in parallel and this mathematical problem can therefore be solved using parallel computing.

### 5. Speed-ups reported

Increased computing speed is the ultimate objective of parallel computing. Table 1 lists the speed-ups that were reported by the parallel computing applications reviewed in this paper.
A speed-up is defined as the ratio of serial computing time divided by parallel computing time. Note that not all reviewed publications in this paper have clearly reported speed-ups. And some of the speed-ups were determined by the authors from graphs or conversions of computing times that were reported from the publications. Table 1 also lists the number of computing units that have been used in the corresponding publication as well as the efficiency. As discussed in the introductory section, a computing unit can be any of the items in the following name list: computer, node, CPU, core, process or thread. The efficiency is then defined as the ratio of the speed-up divided by the number of computing units. Multiple speed-ups can be reported from the same application using different

| References                     | Computing unit | Speed-up | Efficiency (%) | Resource type |
|--------------------------------|----------------|----------|----------------|---------------|
| Wu et al. 2017a [40]           | 216            | 208      | 96             | Cluster       |
| Zawidzki and Szklarski [58]    | 152(64)        | 120(60)  | 79(94)         | PC/GPU        |
| Wu et al. 2018c [39]           | 56             | 55.3     | 99             | Cluster       |
| Wu et al. 2018a [42]           | 56             | 54.6     | 97             | Cluster       |
| Wu et al. 2016a [37]           | 56             | 53.6     | 96             | Cluster       |
| Caiño-Lores et al. [63]        | 128(16)        | 14.0(3.3)| 11.21          | Cluster       |
| Zhang et al. [23]              | 16(2)          | 11.0(1.95)| 69(98)        | Cluster       |
| Zhang et al. [26]              | 32(2)          | 9.5(1.95)| 30(70)        | Cluster       |
| Vollebregt [44]                | 8              | 7.4      | 93             | PC            |
| Fung et al. [49]               | 8              | 7.3      | 91             | Cluster       |
| Wu et al. 2017b [41]           | 8              | 7.2      | 90             | Cluster       |
| Zhang et al. [84]              | 8              | 7.1      | 89             | PC            |
| Firuziyan and von Estorff [22] | 10(2)          | 6.5(1.96)| 65(98)        | Cluster       |
| Iqbal et al. [52]              | 8              | 6.4      | 80             | PC            |
| Iqbal et al. [51]              | 8              | 6.3      | 79             | PC            |
| Xiao et al. [7]                | 8(2)           | 6.1(1.84)| 80(92)        | Cluster       |
| Liang et al. [70]              | 8(2)           | 5.5(1.8) | 69(90)        | Cluster       |
| Eberhard et al. [12]           | 9              | 3.8      | 42             | Cluster       |
| Pogorelov et al. [15]          | 4              | 3.4      | 85             | PC            |
| Fung et al. [10]               | 4              | 3.4      | 85             | ALU           |
| Ding et al. [9]                | 4              | 3.4      | 85             | ALU           |
| Hoang et al. [24]              | 4              | 2.5      | 63             | PC            |
| Wang et al. [68]               | 5              | 2.5      | 50             | Cluster       |
| Horas et al. [55]              | 4              | 2.5      | 62             | PC            |
| Hoang et al. [19]              | 4(2)           | 2.4(1.75)| 60(87)        | PC            |
| Sugiyama et al. [82]           | 4              | 2.0      | 50             | PC            |
| Wu et al. 2018b [32]           | 3              | 1.9      | 62             | PC            |
| Xun et al. [69]                | 16(8)          | 1.8(1.4)| 11(18)        | Cluster       |
| Calderon et al. [35]           | 4              | 1.8      | 45             | PC            |
| Matsuura and Miyatake [56]     | 4              | 1.7      | 42             | PC            |
| Wu and Cole [13]               | 2              | 1.43     | 72             | PC            |
numbers of computing units. Considering the load balance issue of parallel computing and communication cost as discussed in Reference [13], the case with the highest speed-up may not have the highest efficiency. There is also the scalability issue that has been discussed by Fernandez et al. [81]. For publications that have two pairs of parameters, the first set of values (outside of the brackets) indicates the case that has the highest speed-up while the second set of values (in the brackets) indicates the case that has the highest efficiency. Table 1 also lists the computing resource types that were used for the applications. The following points should be noted for the computing resource types: (1) Zawidzki and Szklarski [58] used GPUs as the computing units while all others in the table used CPUs; (2) parallelised computing units in References [9,10] are ALUs which are components of traditional CPUs; (3) applications in References [68–70] used the MapReduce technique which is mostly cloud based, but non-commercial clusters were used in these applications; (4) the cluster used in Reference [63] is cloud based which has the nature of Infrastructure as a service; and (5) all other clusters in the table except Reference [63] indicated use of the conventional form of clusters that are only available for individuals or a group.

About half of the applications listed in Table 1 have used PCs and high parallel computing efficiencies were achieved. For example, applications in References [44,49,58] have reported parallel computing cases with efficiencies over 90%. Further, all parallel computing cases using PCs have 50% efficiency or higher except References [35] and [56] with 45% and 42% respectively. Comparatively, parallel computing using clusters can be more powerful in terms of speed-ups. Among the top ten speed-ups listed in Table 1, seven of them were achieve using clusters; and the ones achieved by using clusters are significantly higher than the ones using PCs due to the fact that significantly more computing units are available from clusters. Another interesting observation that can be made from Table 1 is that nine applications using clusters have reported parallel computing efficiencies of 90% or higher, which is more than the corresponding number of applications (three) using PCs. Also, only three applications [12,63,69] using clusters have reported parallel computing efficiencies lower than 50%. Note that this can be a mere coincidence as there is no clear justification for PCs to have lower parallel computing efficiency than clusters.

In addition to the specific speed-ups reported from the publications, other benefits of parallel computing can be variously described as to achieve: (1) real-time hardware-in-loop applications [61,62,74]; (2) real-time monitoring [60]; (3) higher system capacity [59]; (4) higher model flexibility for extensions [31]; and (5) better optimisation results [8].

### 6. Challenges for applying parallel computing

Figure 12 shows the number of parallel computing publications in railway research by years. The first publication found was by Holmes et al. [27] who studied the CFD issues regarding train-tunnel entry flow. It can be seen that application of parallel computing in railway research has only 20 years of history. This is a short history compared with some other industries like aviation. Parallel computing has been applied by the US National Aeronautics and Space Administration since the 1970s [86]. However, the short history of parallel computing applications in railway research is also not surprising considering the following facts: (1) the engineering basis using clusters for parallel computing was arguably invented by Gene Amdahl in 1967 [6,14]; (2) PCs were only made available in the late 1970s; and (3) the MPI and OpenMP standards were only released in 1994 and 1997.
respectively [34,46]. Figure 12 also shows that there is a fluctuating increasing number of publications over the years. However, the publication number to date is still very low, being less than 10 publications per year. Upon completing the literature review of this paper in September 2018, nine publications were found in 2018. It can be seen that there is significant room for leveraging the applications of parallel computing in railway research.

The following contents of this section discuss the challenges of applying parallel computing from two perspectives: (1) time and computer resource investment versus speed-ups; and (2) relatability and flexibility of the parallelised codes. The discussions can be generally applied to many research fields including railway research.

### 6.1. Time and computer resource investment versus speed-ups

Whenever a researcher is encouraged to use parallel computing, a factor that must be considered is the balance between the complexity of applying parallel computing (time investment) and the speed-ups that can be achieved. Regarding the time investment, the authors want to point out that, due to the continuous development of parallel computing enabling techniques such as MPI, OpenMP and Pthreads, the initialisation processes to programme parallelised codes have been minimised. For example, a very simple parallelised code enabled by OpenMP is shown in Figure 13. The initialisation process can be more complicated in practical engineering codes, but this aspect should not become the barrier that is preventing researchers from using parallel computing.

For the case of ready-to-use commercial software packages, the time investment is even lower. Therefore, the true balance between the time investment and speed-ups should be determined by the studied question itself, i.e. the independence level and load balance of the parallelised computing tasks.

As explained previously, the independence level of a computing task indicates how much communications are needed with other computing tasks during the simulation. Specifically, the higher the independence level, the less the amount of communications required. Generally, it can be said that higher independence levels result in less time investment and higher speed-ups. With modern parallel computing techniques, programmers’ tasks are mainly to design the partitioning of the code blocks to be executed by different computing units. If the computing tasks have a high independence level, then the code blocks should be able to easily be partitioned and less code lines are needed to connect (communicate) the code blocks. Also, it is understandable that less
communications result in less parallel computing overhead and higher speed-ups. Example applications that have highly independent tasks were found in the literature in the areas of parallel optimisations \[12,39,52,58\], data and signal processing \[65,67,69,71\] and power supply simulations \[35,63,64,75,81\]. Also, as mentioned in Section 3.3, co-simulations related studies \[87\] have naturally independent computation tasks, they are also very suitable for parallel computing.

Load balance is another key factor that determines the speed-ups. A perfect load balance is where all computing units have exactly the same computing loads and all computing units finish their tasks at the same time. In this case, parallel computing can minimise the waiting time for slave units and thus achieve fast computing speed. For railway research, most reported parallel computing applications have naturally well balanced computing loads except some applications regarding VSD simulations \[13,31,54\]. When load balance is not well controlled, parallel computing can result in even longer computing time than serial computing as discussed in Reference \[13\]. In that case, one unbalanced parallel simulation of train dynamics took about 1.7 times longer than the serial simulation.

In addition to previous discussions, another factor that must be considered during parallel computing is the computer resources. As can be seen from Table 1, clusters can usually achieve higher speed-ups than PCs as the former have more computing units. However, clusters are not available to every researcher and the authors believe that most railway researchers are still not very familiar with cluster applications. For researchers who have access to clusters, the authors encourage them to investigate how clusters can be used for their particular research purposes. If parallelisable computing tasks can be easily identified, those computing tasks tend to have high parallelisability and high speed-ups could be achieved without a lot of time investment. Also, clusters have the advantage of a large quantity of computing units; in this case, parallel computing efficiency could become less important and the programming complexity could be lower.

Comparatively, parallel computing applications on PCs have lower speed-ups due to the limited number of computing units on PCs. But PCs require much less investment and are more flexible and convenient to use. Nowadays most PCs have no less than four computing cores; proper utilisation of parallel computing on PCs can also provide considerable speed-ups. For example, Vollebregt \[44\] achieved a 7.4 speed-up using an eight-core PC; Pogorelov et al. \[15\] achieved a 3.4 speed-up using a four-core PC; and Wu and Cole \[13\] achieved a 1.43 speed-up using only two computing cores.

6.2. Reliability and flexibility of the parallelised codes

Having been able to programme for parallel computing, two other factors that also need to be paid attention are reliability and flexibility. Compared with conventional serial...
codes, the reliability issue regarding parallel codes is mainly about the race condition. The race condition refers to the fact that, once a parallelised computing task is commenced, the computing unit will try to finish that task as soon as possible [6]. Without proper computing process control, several bugs that do not exist in serial computing could be caused. For example, two computing units may try to write to the same memory address (the same shared parameter) and different computing units may write to and read the same memory address in a random sequence rather than a specific sequence as required. Despite the existence of the race condition, it can be well controlled by using good synchronisation methodologies [6] and special programming techniques [45].

The flexibility issue implies the scalability of the code and the adoptability of the code on serial computers. Scalability indicates the consistency of the parallel computing efficiency for different numbers of parallelised computing units. As can be seen in Table 1, in the 9 particular applications in which scalability can be assessed, the higher parallel computing efficiency was most commonly achieved using less computing units. High scalability applications refer to the ones that can achieve the same or similar efficiencies when using different numbers of computing units. The scalability of a parallel computing application is mostly related to its independence level as discussed in Section 6.1. The higher the independence level, the less the communications required which usually results in higher parallel computing efficiency. One of the most scalable applications are the iterative optimisations [12,39,52,58]; this is mainly contributed to by the fact that the number of individuals during the same iteration can be easily adjusted according to the available slave computing units. Other applications such as data processing [65,67,69,71] and power supply simulations [35,63,64,75,81] also have good scalability, mainly due to the fact that they also have homogeneous and highly independent computing tasks. The adoptability issue is mainly about decreasing the workload for code maintenance. Ideally, researchers want parallelised code that can achieve the same function on serial computers as well, so only one piece of code needs to be maintained. This has now become achievable using most parallel computing enabling techniques. For example, OpenMP has specially formatted syntax [46] which allows serial compilers to function as usual when compiling a parallelised code. However, special attention has to be paid during the programming if adoptability is needed. For example, when coding an iterative optimisation problem, the partitioning of the fitness assessment has to be done by one group of individuals per computing unit instead of one individual per computing unit. The group can have one individual or multiple individuals, which can be adjusted according to the number of available computing units. In this scenario, some applications have naturally good adoptability such as FEA, data and signal processing and power supply simulations; these applications always have multiple elements as a group and the partitioning is always done in the fashion of one group per computing unit. When the number of available computing units varies, researchers just need to adjust the size of the group accordingly.

7. Conclusions

In this paper, parallel computing applications and enabling techniques that are used in railway research were reviewed. Nine enabling techniques were reviewed and MPI, DDM
and Hadoop & Apache are the top three most widely used enabling techniques. Seven major application topics were reviewed and iterative optimisations, FEA/CFD/DEA and Data and signal analysis are the most widely reported applications. The reasons why these applications are suitable for parallel computing were discussed as well as the suitability of the various enabling techniques for different applications.

About half of the applications used PCs while the others used clusters. Comparatively, parallel computing applications on PCs have lower speed-ups due to the limited number of computing units on PCs. However, PCs require much less investment and are more flexible and convenient to use. Among the top ten speed-ups found in this paper, seven of them were achieved using clusters and those achieved by using clusters are significantly higher than the ones achieved using PCs.

The challenges for conducting parallel computing in railway research is to formulate highly independent parallelisable computing tasks and assign balanced computing loads to the computing units. The race condition exists in almost all parallel computing programming, but it can be well controlled by using good synchronisation methodologies and special programming techniques. Iterative optimisations, data and signal processing and power supply simulations are three good examples of applications that have highly independent computing tasks that can have good scalability and flexibility when using parallel computing.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**References**

[1] Top500: performance development – exponential growth of supercomputing power [Internet]. Mannheim, Germany: Prometeus GmbH; c1993–2018 [cited 2018 Sep 8]. Available from: https://www.top500.org/statistics/perfdevel/

[2] Rupp K. 42 Years of microprocessor trend data [Internet]. [cited 2018 Nov 22]. Available from: https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/

[3] USBLC. Recent price trends in the computer and peripheral industry. Washington (DC): US Bureau of Labor Statistics; 2017.

[4] Amazon Web Services: AWS & cloud computing [Internet]. c2018 [cited 2018 Sep 8]. Available from: https://aws.amazon.com/?nc2=h_lg

[5] Google Cloud: One cloud, endless possibilities [Internet]. [cited 2018 Sep 8]. Available from: https://cloud.google.com/

[6] Pacheco P. An introduction to parallel programming. Burlington (MA): Morgan Kaufmann; 2011.

[7] Xiao Q, Wang C, Guo G. The research of parallel computing for large-scale finite element model of wheel/rail rolling contact. 3rd IEEE International Conference on Computer Science and Information Technology; 2010 Jul 9–11; Chengdu, China.

[8] Boman S, Gerdovci P. Re-scheduling the railway traffic using parallel simulated annealing and Tabu search: a comparative study [Master of Science Thesis]. Karlskrona, Sweden: Blekinge Institute of Technology; 2015.

[9] Ding Y, Zhou F, Bai Y, et al. Parallel computing for multi-train movement simulation on electrified railway. Second International Conference on Information and Computing Science; 2009 May 21–22; Manchester, UK.
[10] Fung YF, Ho TK, Cheung WL. Parallel solution for railway power network simulation. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing; 2001 Aug 26–28; Victoria, Canada.

[11] Kogut J, Ciurej H. A vehicle-track-soil dynamic interaction problem in sequential and parallel formulation. Int J Appl Math Comput Sci. 2010;20(2):295–303.

[12] Eberhard P, Dignath F, Kubler L. Parallel evolutionary optimization of multibody systems with application to railway dynamics. Multibody Sys Dyn. 2003;9(2):143–164.

[13] Wu Q, Cole C. Computing schemes for longitudinal train dynamics: sequential, parallel and hybrid. J Comput Nonlin Dyn. 2015;10(6):064502.

[14] Rauber T, Runger G. Parallel programming for multicore and cluster systems. Heidelberg, Germany: Springer; 2010.

[15] Pogorelov D, Yazykov V, Lysikov N, et al. Train 3D the technique for inclusion of three dimensional models in longitudinal train dynamics and its application in derailment studies and train. Veh Syst Dyn. 2017;55(4):583–600.

[16] Zhao J. Research on framework for urban railway massive data. 4th IEEE International Conference on Information Science and Technology; 2014 Apr 26–28; Shenzhen, China.

[17] Wang Y, Zhou W, Xie D, et al. Automatic visual inspection for catenary on high-speed railways. First International Conference on Rail Transportation; 2017 Jul 10–12; Chengdu, China.

[18] Diedrichs B. On computational fluid dynamics modelling of crosswind effects for high-speed rolling stock. J Rail Rapid Transit. 2003;217(3):203–226.

[19] Hoang T, Alart P, Dureisseix D, et al. A domain decomposition method for granular dynamics using discrete elements and application to railway ballast. Ann Solid Struct Mech. 2011;2(2–4):87–98.

[20] Dolean V, Jolivet P, Nataf F. An introduction to domain decomposition methods: algorithms, theory and parallel implementation. Philadelphia (PA): Society for Industrial and Applied Mechanics; 2016.

[21] Yang X, Wang H, Jin X. Numerical analysis of a train-bridge system subjected to earthquake and running safety evaluation of moving train. Shock Vib. 2016;2016:9027054.

[22] Firuziaan M, von Estorff O. Simulation of the dynamic behavior of bedding-foundation-soil in the time domain. Lect Notes Appl Mech. 2003;6:357–376.

[23] Zhang X, Jin X, Chen X. Simulation of the interactions between a train and a long-span cable-stayed bridge using parallel computing with domain decomposition. J Rail Rapid Transit. 2012;226(4):347–359.

[24] Hoang T, Saussine G, Dureisseix D, et al. Domain decomposition with discrete element simulations using shared memory parallel computing for railways applications. Eur J Comput Mech. 2012;21(3–6):242–253.

[25] Du X, Jin X, Chen X. Dynamic analysis of steel plate-truss composite girder on long span two-layer cable-stayed bridge [in Chinese]. J Civil Archit Environ Eng. 2010;32(2):82–86.

[26] Zhang W, Jin X, Gao L, et al. Dynamic response analysis for a highway-railway double-duty tunnel using parallel computing [in Chinese]. J Vibr Shock. 2012;31(8):164–170.

[27] Holmes B, Dias J, Rifai S, et al. Solution of train-tunnel entry flow using parallel computing. Comput Mech. 1999;23(2):124–129.

[28] Sakai H, Takagaki M, Hayashi M, et al. Analysis of rolling contact behaviour between wheel and rail through large-scale parallel computing. Q Rev Railway Tech Res Inst (Japan). 2014;55(3):171–175.

[29] Akama M, Sasakura M, Furuno K. Design and analysis of low-stress and low-noise light-weight railway wheel. J Rail Rapid Transit. 2009;223(1):45–59.

[30] Yaskawa S. Bullet train operation simulator including electric power consumption. International Congress on Modelling and Simulation Proceedings (MODSIM01); 2001 Dec 10–13; Canberra, Australia.

[31] Wu Q, Sun Y, Spiryagin M, et al. Parallel co-simulation method for railway vehicle-track dynamics. J Comput Nonlin Dyn. 2018;13(4):041004.
[32] Wu Q, Spiryagin M, Cole C, et al. Introducing wheel-rail adhesion control into longitudinal train dynamics [Internet]. Int J Heavy Veh Syst. 2018. In press. Available from: http://www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijhvs

[33] Mrzyglod M, Oszyczka A. Optimization of railway vehicle structures using evolutionary algorithms and parallel computing technologies. 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference; 2006 Sep 6–8; Portsmouth, VA.

[34] Barney B. Message Passing Interface (MPI) [Internet]. Livermore (CA): Lawrence Livermore National Laboratory; [updated 2018 Jun 29 00:36:08; cited 2018 Sep 8]. Available from: https://computing.llnl.gov/tutorials/mpi/

[35] Calderon A, Garcia A, Garcia-Carballeira F, et al. Improving performance using computational compression through memorization: a case study using a railway power consumption simulator. Int J High Perform Comput Appl. 2016;30(4):469–485.

[36] Radio VC. Optimization study of dynamic vibration absorbers parameters and distribution for its application on railway tunnels for the reduction of railway-induced vibration [Report]. Barcelona, Spain: Polytechnic University of Catalonia; 2015.

[37] Wu Q, Cole C, Spiryagin M. Parallel computing enables whole-trip train dynamics optimizations. J Comput Nonlin Dyn. 2016;11(4):044503.

[38] Wu Q, Cole C, McSweeney T. Applications of particle swarm optimization in the railway domain. Int J Rail Trans. 2016;4(3):167–190.

[39] Wu Q, Cole C, Spiryagin M, et al. Parallel multiobjective optimisations of draft gear designs. J Rail Rapid Transit. 2018;232(3):744–758.

[40] Wu Q, Spiryagin M, Cole C. Parallel computing scheme for three-dimensional long train system dynamics. J Comput Nonlin Dyn. 2017;12(4):044502.

[41] Wu Q, Cole C, Spiryagin M, et al. railway air brake model and parallel computing scheme. J Comput Nonlin Dyn. 2017;12(5):051017.

[42] Wu Q, Sun Y, Spiryagin M, et al. Methodology to optimize wedge suspensions of three-piece bogies of railway vehicles. J Vibr Control. 2018;24(3):565–581.

[43] Bettinelli A, Santini A, Vigo D. A real-time conflict solution algorithm for the train rescheduling problem. Transp Res Part B Methodol. 2017;106(C):237–265.

[44] Vollebregt EAH. Refinement of Kalker’s rolling contact mode. 8th International Conference on Contact Mechanics and Wear of Rail/Wheel Systems (CM2009); 2009 Sep 15–18; Firenze, Italy.

[45] Duarte OM, Tummelshammer P. Deterministic parallel programming for railway applications. EMC² Summit; 2016 Apr 11; Vienna, Austria.

[46] Barney B. OpenMP [Internet]. Livermore (CA): Lawrence Livermore National Laboratory; [updated: 2018 Jun 07 09:54:37; cited 2018 Sep 8]. Available from: https://computing.llnl.gov/tutorials/openMP/

[47] Barney B. POSIX threads programming [Internet]. Livermore (CA): Lawrence Livermore National Laboratory; [updated: 2017 Mar 08 04:24:00; cited 2018 Sep 8]. Available from: https://computing.llnl.gov/tutorials/pthreads/

[48] Beveridge J, Wiener R. Multithreading applications in Win32: the complete guide to threads. Boston (MA): Addison-Wesley; 1997.

[49] Fung Y, Ho T, Cheung W. Real-time simulation for power system based on parallel computing- an empirical study. 5th International Conference on Advances in Power System Control, Operation and Management, APSCOM 2000; 2000 Oct–Nov 30–1; Hong Kong, China.

[50] Bosomworth C, Spiryagin M, Alahakoon S, et al. The influence of vehicle system dynamics on rail foot heat transfer. Aust J Mech Eng. 2018;16(2):126–138.

[51] Iqbal S, Grahn H, Kräsemann J. A parallel DFS algorithm for train re-scheduling during traffic disturbances - early results. 4th Swedish Workshop on Multicore Computing MCC; 2011 Nov 23–25; Linköping, Sweden.

[52] Iqbal SMZ, Grahn H, Kräsemann JT. A parallel heuristic for fast train dispatching during railway traffic disturbances - early results. 1st International Conference on Operations Research and Enterprise Systems (ICORES-2012); 2012 Feb 4–6; Vilamoura, Portugal.
[53] Xie Y, Li X, Liu F. A simple and efficient parallel approach to large-scale railway freight data analysis. 12th IEEE International Conference on High Performance Computing and Communications; 2010 Sep 1–3; Melbourne, Australia.

[54] Al-Oraiqat A. Parallel implementation of a vehicle rail dynamical model for multi-core systems. Int J Adv Stud Comput Sci Eng. 2017;6(1):34–41.

[55] Horas CS, Alencar G, De Jesus AMP, et al. Development of an efficient approach for fatigue crack initiation and propagation analysis of bridge critical details using the modal superposition technique. Eng Fail Anal. 2018;89:118–137.

[56] Matsuura G, Miyatake M. Optimal train speed profiles by dynamic programming with parallel computing and the fine-tuning of mesh. WIT Trans Built Environ. 2014;135:767–777.

[57] Santur Y, Karakose M, Akin E. An adaptive fault diagnosis approach using pipeline implementation for railway inspection. Tur J Electr Eng Comput Sci. 2018;26(2):987–998.

[58] Zawidzki M, Szklarski J. Effective multi-objective discrete optimization of Truss-Z layouts using a GPU. Appl Soft Comput. 2018;70:501–512.

[59] Zhang Z. Development of an FPGA system for parallel processing of railway non-destructive testing data [Master of Research Thesis]. Birmingham (UK): University of Birmingham; 2014.

[60] Khan J, Tatkeu C, Deloof P. Multiple sensor automatic train control system implementation in FPGA. The 9th World Congress on Railway Research; 2011 May 22–26; Lille, France.

[61] Zhou YJ, Mei TX, Freear S. FPGA implementation of wheel-rail contact laws. UKACC International Conference on Control; 2008 Sep 2–4; Manchester, UK.

[62] Zhou YJ, Mei TX, Freear S. Field programmable gate array implementation of wheel-rail contact laws. IET Control Theory Appl. 2010;4(2):303–313.

[63] Caíno-Lores S, Fernández AG, García-Carballeira F, et al. A cloudification methodology for multidimensional analysis Implementation and application to a railway power simulator. Simul Modell Pract Theory. 2015;55:46–62.

[64] Caíno-Lores S, García A, García-Carballeira F, et al. Efficient design assessment in the railway electric infrastructure domain using cloud computing. Integr Comput Aided Eng. 2016;24(1):1–16.

[65] Cao Y, Li P, Zhang Y. Parallel processing algorithm for railway signal fault diagnosis data based on cloud computing. Future Gener Comput Syst. 2018;88:279–283.

[66] Shao Y, Liu R, Wang F, et al. Research on big data management for high-speed railway equipment. Appl Mech Mater. 2013;462-463:405–409.

[67] Xiao Y, Cheng Y, Fang Y. Research on cloud computing and its application in big data. Chem Eng Trans. 2015;46:325–330.

[68] Wang Z, Li T, Zhang J, et al. Parallel method for pre-processing high-speed railway noise data based on MapReduce. J Commun. 2011;32(9A):263–269.

[69] Xun W, An Y, Jie R. Application of parallel particle swarm optimize support vector machine model based on Hadoop framework in the analysis of railway passenger flow data in China. Chem Eng Trans. 2015;46:367–372.

[70] Liang Y, Du C, Liu J, et al. Research on parallel processing method for ground penetrating radar data in state detection of railway subgrade [in Chinese]. China Railway Sci. 2017;38(2):11–18.

[71] Sangat P, Indrawan-Santiago M, Taniar D, et al. Processing high-volume geospatial data: a case of monitoring heavy haul railway operations. Procedia Comput Sci. 2016;80:2221–2225.

[72] Oneto L, Fumeo E, Clerico G, et al. Dynamic delay predictions for large-scale railway networks: deep and shallow extreme learning machines tuned via Thresholdout. IEEE Trans Syst Man Cybern Syst. 2017;47(10):2754–2767.

[73] Oneto L, Fumeo E, Clerico G, et al. Train delay prediction systems: a big data analytics perspective. Big Data Res. 2018;11:54–64.

[74] Bosso N, Zampieri N. Using multi-core architectures to solve the wheel-rail contact problem. Int J Appl Eng Res. 2017;12(20):9592–9604.
[75] Desjouis B, Remy G, Ossart F, et al. A new generic problem formulation dedicated to electrified railway systems. International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS); 2015 Mar 3–5; Aachen, Germany.

[76] Huang H, Li K, Schonfeld P. Real-time energy-saving metro train rescheduling with primary delay identification. PLoS ONE. 2018;13(2):e0192792.

[77] Monnoyer F, Lorriaux E, Bourabaa N. Aerodynamic optimization of railway motor coaches. 7th World Congress on Railway Research; 2006 Jun 4–8; Montréal, Canada.

[78] Takaishi T. Parallel computing of aerodynamic noise from a complex-shaped body. Trans Jpn Soc Mech Eng Ser B. 2012;78(790):1206–1219.

[79] Lockwood GK. Conceptual overview of map-reduce and hadoop [Internet]. Berkeley (CA): Glen K. Lockwood, PhD; [updated: 2015 Oct 9 at 11:17 AM PDT; cited 2018 Sep 9]. Available from: https://www.glennklockwood.com/data-intensive/hadoop/overview.html

[80] Liu Z, Dong C, Pei Y, et al. Online rail straightness measurement based on parallel computing. 3rd International Conference on Material, Mechanical and Manufacturing Engineering (IC3ME); 2015 Jun 27–28; Guangzhou, China.

[81] Fernandez AG, Caino-Lores S, Serrano E, et al. Scalable data management techniques for high-end computing systems: technical report on application requirements and use cases. Madrid, Spain: Universidad Carlos III de Madrid; 2017. (Technical Report E5.1).

[82] Sugiyama H, Yamashita S, Suda Y. Curving simulation of ultralow-floor light rail vehicles with independently rotating wheelsets. ASME 2010 International Mechanical Engineering Congress and Exposition; 2010 Nov 12–18; Vancouver, Canada.

[83] Vollebregt EAH User guide for CONTACT, rolling and sliding contact with friction. Delft, The Netherlands: VORtech CMCC; 2018. (Technical Report TR09-03).

[84] Zhang J, Zhao Y, Zhang Y, et al. Non-stationary random vibration of a coupled vehicle–slab track system using a parallel algorithm based on the pseudo excitation method. J Rail Rapid Transit. 2013;227(3):203–216.

[85] Abril M, Salido MA, Barber F. Application of meta-tree-based distributed search to the railway scheduling problem. CP/ICAPS 2007 joint Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems; 2007 Sep 23; Providence, RI.

[86] Bjurman BE, Jenkins GM, Masreliez CJ, et al. Airborne advanced reconfigurable computer system. WA: Boeing Commercial Airplane Company; 1976. (Seattle, NASA Contractor Report 145024).

[87] Spiryagin M, Persson I, Wu Q, et al. A co-simulation approach for heavy haul long distance locomotive-track simulation studies. Veh Syst Dyn. 2018;1–18. DOI:10.1080/00423114.2018.1504088