Review

Data Locality in High Performance Computing, Big Data, and Converged Systems: An Analysis of the Cutting Edge and a Future System Architecture

Sardar Usman [1], Rashid Mehmood [2,] *, Iyad Katib [3] and Aiiad Albeshri [3]

[1] Department of Computer Science, Grand Asian University Sialkot, Pakistan; sardar.usman@gauss.edu.pk; (S.U)
[2] High Performance Computing Center, King Abdulaziz University, Jeddah, 21589, Saudi Arabia; RMehmood@kau.edu.sa; (R.M.)
[3] Department of Computer Science, FCIT, King Abdul Aziz University, Jeddah, 21589, Saudi Arabia; iakatib@kau.edu.sa, aaalbeshri@kau.edu.sa; (I.K.), (A.A)
* Correspondence: RMehmood@kau.edu.sa; (R.M.)

Abstract: Big data has revolutionised science and technology leading to the transformation of our societies. High Performance Computing (HPC) provides the necessary computational power for big data analysis using artificial intelligence and methods. Traditionally HPC and big data had focussed on different problem domains and had grown into two different ecosystems. Efforts have been underway for the last few years on bringing the best of both paradigms into HPC and big converged architectures. Designing HPC and big data converged systems is a hard task requiring careful placement of data, analytics, and other computational tasks such that the desired performance is achieved with the least amount of resources. Energy efficiency has become the biggest hurdle in the realisation of HPC, big data, and converged systems capable of delivering exascale and beyond performance. Data locality is a key parameter of HPDA system design as moving even a byte costs heavily both in time and energy with an increase in the size of the system. Performance in terms of time and energy are the most important factors for users, particularly energy, due to it being the major hurdle in high performance system design and the increasing focus on green energy systems due to environmental sustainability. Data locality is a broad term that encapsulates different aspects including bringing computations to data, minimizing data movement by efficient exploitation of cache hierarchies, reducing intra- and inter-node communications, locality-aware process and thread mapping, and in-situ and in-transit data analysis. This paper provides an extensive review of the cutting-edge on data locality in HPC, big data, and converged systems. We review the literature on data locality in HPC, big data, and converged environments and discuss challenges, opportunities, and future directions. Subsequently, using the knowledge gained from this extensive review, we propose a system architecture for future HPC and big data converged systems. To the best of our knowledge, there is no such review on data locality in converged HPC and big data systems.

Keywords: High Performance Computing (HPC); big data; High Performance Data Analytics (HPDS); convergence; data locality; spark; Hadoop; design patterns; process mapping; in-situ data analysis

1. Introduction

Data has grown exponentially during the last decade giving rise to the big data phenomenon [1], [2]. Big data has revolutionised science and technology leading to innovations in many sectors including urbanisation [3], transport [4], energy [5], healthcare [6], education [7], economics [8], smart societies [9], computing infrastructure [10], and more; see e.g., [1], [11], for details on big data technologies and applications. The paramount contribution of big data is the development of contemporary data-driven machine and

 doi:10.20944/preprints202211.0161.v1

deep learning and artificial intelligence (AI) technologies that have transformed our societies and infrastructure [12], [13]. This has also given rise to the need for developing green AI approaches, a broad term that incorporates properties including energy efficiency, responsibility, fairness, etc. [14]–[17].

High Performance Computing (HPC) provides the necessary computational power for big data analysis using machine learning and other artificial intelligence methods [18]. HPC has traditionally focussed on compute-intensive simulations of natural phenomena, engineering design of static and dynamic objects (bridges, vehicles, etc.), and other scientific and engineering problems on high-end tightly-coupled supercomputing systems. Big data technologies have grown to use relatively loosely-coupled and inexpensive computer systems. HPC is good at compute-intensive tasks while big data systems have better performance for data-driven tasks. The last five or so years have seen developments of systems that integrate the advantages of both big data and HPC systems by converging the two approaches of system design [19]–[26].

Designing HPC and big data converged systems, also referred to as High Performance Data Analytics (HPDA), is a hard task requiring careful placement of data, analytics, and other computational tasks such that the desired performance is achieved with the least amount of resources. Energy efficiency has become the biggest hurdle in the realisation of HPC and HPDA systems capable of delivering exascale and beyond performance. In large-scale systems comprising millions of cores and thousands of nodes aiming to provide exascale performance is an arduous exercise. The classical Dennard scaling has stopped more than a decade now as the scaling of a single processing core is ceased and performance scaling is achieved by mounting more cores on chips and exploiting explicit parallelism. The complexity of managing parallelism increases with increasing memory hierarchies from system and node levels to the processing unit level [27]. It becomes even more challenging in loosely coupled systems where nodes are typically geographically distributed with many uncertainties including network quality [10], [28], [29].

Locality, or how to improve data access and transfer, within the application, is one of the most significant challenges that will need to be addressed in the upcoming years. Addressing data locality issues in simple terms means narrowing the distance between data and processing for better performance. One problem relating to locality comes from the memory and the network: the affinity and location of processes within an application affect how quickly data is transferred between them. The cost of data movement has been under the scanner of researchers for years but now has gained momentum, as performance and energy consumption are heavily dependent on data locality. Researchers have realised that scalability cannot be addressed only by powerful infrastructure but actually constrained by resource utilization and energy efficiency [27].

Locality refers to a phenomenon in that computations do not have uniform or independent access to data but rather have clustered, dependent, co-related access [30]. **Principle of Locality or Locality of References** can be categorized as temporal locality and spatial locality. Temporal locality aims to reuse data as much as possible once it has been brought in and spatial locality aims to use every data element that has been brought in [31]. **Data Locality** can be defined as bringing computations to the place (processor, cache, etc.) or nodes where the data to be processed actually resides. **Data Movement** is defined as the movement of data across cache hierarchies, inter- and intra-node data movement and, in the case of in-situ data analysis, data movement mostly refers to data movement back and forth to persistent storage and retrieving data for post-data-analysis. In the rest of the paper, we refer to data locality as a means of bringing computation to data, minimizing data movement by efficient exploitation of cache hierarchies, reducing intra- and inter-node communications, locality-aware process and thread mapping, and in-situ and in-transit data analysis.

Data locality is a key parameter of HPDA system design as moving even a byte costs heavily both in time and energy with an increase in the size of the system. Performance in terms of time and energy are the most important factors for users, particularly energy,

 doi:10.20944/preprints202211.0161.v1

due to it being the major hurdle in high performance system design and the increasing focus on green energy systems due to environmental sustainability.

This paper provides an extensive review of the cutting-edge on data locality in HPC, big data, and converged systems. To the best of our knowledge, there is no such review on data locality in converged HPC and big data systems.

In Section 2, we review earlier works on data locality surveys in HPC, big data, and converged systems and establish the case for this work. Section 3 review literature on data locality in HPC environments with topics covering applications perspective; programming languages, compiler and libraries; cache optimization techniques; locality aware scheduling and load-balancing; bulk synchronous processing (BSP); out-of-core computing; parallelism mapping; and in-situ data analysis. Section 4 reviews works on data locality in HPC environments from perspectives including parallel programming models, data placement, scheduling and load balancing, and in-memory computations. Section 5 reviews literature on data locality in converged HPC and big data environments with topics covering MPI with map-reduce frameworks, map-reduce frameworks with high performance interconnects, and map-reduce-like frameworks for in situ analysis. Section 6 discusses challenges, opportunities, and future directions covering programming paradigms, programming models and language support, programming abstractions, innovations in data layout strategies, locality-aware scheduling, software hardware co-design, and innovations in memory and storage technologies. In Section 7, we use the knowledge gained from this extensive review to propose a system architecture for future converged systems. Section 8 concludes the paper.

2. Related Works: Case for This Paper

This section reviews earlier works on data locality surveys in HPC (Section 2.1), big data (Section 2.2), and converged systems (Section 2.3) and establish the case for this work.

2.1. Big Data

Lores et al. [32] presented a survey of different techniques proposed to deal with data locality for high performance and high throughput systems by categorizing different techniques into four major categories i.e. application development, in-memory computing, task scheduling, and storage formats. A survey by Zhang et al. [33] focused on data processing and management strategies for in-memory computations. Dolev et al. [34] investigate different requirements and challenges in designing geographically distributed big data analysis frameworks and protocols by classifying and focusing on map-reduce-based systems, stream processing, SQL-style processing, geo-distributed frameworks, etc. Senthilkumar et al. [35] base their work on task scheduling in big data computations primarily focusing on scheduler classification, and algorithmic comparison with pros and cons, and also includes various tools and frameworks for managing and enhancing the performance of map-reduce.

Idris et al. [36] presented a survey on context-aware scheduling in the map-reduce frameworks. They classified scheduling techniques and algorithms and comparative analysis of these techniques. Mozakka et al. [37] presented a survey on adaptive job schedulers in map-reduce and discussed the benefits and drawbacks of different adaptive scheduling techniques. Nagina et al. [38] reviewed scheduling algorithms in big data. Sreedhar et al. [39] surveyed big data management and job scheduling. Akilandeswari et al. [40] presented a survey on task scheduling in cloud environments.

2.2. HPC

Hoefler et al. [41] presented an overview of topology mapping focusing on algorithmic strategies and mapping enforcement techniques i.e. resource binding, rank reordering, etc. Unat et al. [27] presented a comprehensive survey of different trends in data locality abstractions available in the form of data structures, runtime systems, libraries and languages and identified opportunities to combine different techniques to address data

 doi:10.20944/preprints202211.0161.v1

locality issues for future HPC systems. Singh et al. [42] presented a comprehensive survey on mapping methodologies by categorizing them as design time, runtime, on-the-fly, and hybrid techniques and also provided upcoming trends, issues, and open challenges for many- and multi-core systems.

2.3. HPC and Big Data Convergence

While the motivations and benefits for the convergence of HPC and big data has been noted in the literature, there are very few papers that have reviewed the literature on the convergence of big data and HPC. The most notable and earliest one (published in 2015) is by Reed and Dongara [18]. This is more of an agenda setting article. Asaadi et al. [43] presented a data-supported comparative survey of HPC (MPI, PGAS, OpenMP) and Big data (Spark, Hadoop, etc.) programming interfaces. They also conducted experiments on a series of benchmarks for comparison and discussed the potential benefits, issues, and challenges of convergence of HPC and big data. Jha et al. [44] presented a comparative analysis of HPC and big data paradigms by disusing their relevant features, functionalities, and common implementation characteristics. They also identified differences between HPC and big data software ecosystems and outlined some architectural similarities along with potential opportunities for the inevitable convergence of HPC and big data. Both these [43], [44] are conference papers.

The technical report [45] reviews convergence challenges and issues raised by the split between conventional HPDA and the explosive growth of data in recent years. They addressed and analysed the application workflow level convergence challenges for widely distributed data resources, challenges imposed by converged infrastructure in edge environments (data flow between data centres and network edge and vice-versa), and opportunities for integrated centralized infrastructure. A survey of big data and HPC tracing from parallel programming models to clusters is provided in [46]. Golasowski et al. [47] discuss the convergence by reviewing four EU Horizon 2020 projects. Other discussion works on the convergence include [48]–[50], and a discussion article in the HPCwire magazine [2].

The works discussed above provide general discussions on HPC and big data convergence. To the best of our knowledge, there are no review papers on data locality in converged HPC and big data systems.

3. Data Locality in HPC Environments

HPC has been fundamental in developing many transformative applications [51]–[55]. These HPC applications require careful design of fundamental algorithms, e.g., the seven dwarfs [56]–[61]. Data locality has been coined as a key aspect among others, and is regarded as one of the main challenge for exascale computing endorsed by many technical reports published in the recent years [62], [63], [64], [65]. The organization of memory into banks and NUMA regions, read only memory and multiple cache hierarchies makes efficient data structure and optimization, a complex task. As we are heading towards exascale systems where cost of data movement will be a dominant factor in terms of performance and energy efficiency. The complexities of managing data with different levels of memory hierarchies is further complicated by inter and intra node level communication. Parallelism and communication are further constrained by heterogeneous core architecture. With the increase in platform heterogeneity, portability is a core issue that demands standardization of widely used approaches to automate performance tuning across different architectures [27].

The concurrency and input data size are on a rise and there is need for efficient exploitation of heterogeneous architectures with increasing number of cores to bridge a performance gap between application and target architecture. There is need for optimization of data layout, data movement between processes/threads and data access patterns. The locality issues exist at different levels from how application data is layout to increasing number of processing cores, complex memory hierarchies, inter-node communication,

 doi:10.20944/preprints202211.0161.v1

interconnects and storage units [66]. The following section discusses the research related to data locality in high performance computing domain.

3.1. Application Perspectives

Data locality from application's point of view demands, examining range of modelling methodologies, which needs exploration of huge application space. The adaption of current HPC application code into exascale systems demands efficient utilization of heterogeneous resources, requiring innovations in application layout strategies, communication optimization, synchronization, resource availability, data movement and access strategies.

3.2. Programming Languages, Compiler and Libraries

The increasing level of hardware parallelism and deep hierarchies (core dies, chips, to node level) have posed a challenging task for efficient parallelism and data locality exploitation at all levels of hierarchy which demands data locality support at different levels of software eco system i.e. Programming language, Compiler, Libraries etc. Majo et al. [67] proposed a parallel programming library for composable and portable data locality optimizations for NUMA systems. This library is based on Intel Threading building Blocks (TBB) and allows capturing programmer's insights for mapping tasks to available resources. Lezos et al. [68] presented a compiler directed data locality optimization in MATLAB by developing a software tool MemAssist, for efficient exploitation of targeted architecture and memory hierarchy. Regan-Kelley et al. [69] proposed a language and compiler support for optimizing parallelism, locality and re-computation in image processing pipelines.

Current HPC models lacks efficient exploitation of data locality due lack of language support for data locality policies e.g. array layouts, parallel scheduling etc., over exposing of target architecture and lack of support for user defined policy abstractions. Chapel is a prominent parallel programming language developed by joint efforts of Cray Inc. along with academia and industry with prime focus on separation of parallelism and locality, multi-resolution design with enhanced productivity features [70]. X10 [71] is based on PGAS model and designed specifically for parallel computing that supports structured and unstructured parallelism, user defined primitive struct types and globally distributed arrays with co-location of execution and data. Huang et al. [72] addresses the data locality issues with explicit programming control of locality in the context of OpenMP and how it can be accomplished.

3.3. Cache Optimization Techniques

Locality optimizations by making efficient use of cache have been an active area of research for years. Locality can be categorized as temporal locality (reuse data that brought it into the memory) and spatial locality (use of every data element brought into the memory). The efficient exploitation of registers involves compiler and assembly language programming level optimizations. Cache line length, cache size and cache replacement policy are some of the factors considered for effective and efficient optimization of caches [33]. Gupta et al. [73] proposed spatial locality aware cache partitioning scheme by measuring spatial and temporal locality dynamically for optimal workload block size and capacity for effective cache sharing. Gonzalez et al. [74] proposed a dual data cache organization for managing spatial and temporal locality by implementing lazy cache policy which use locality prediction table to make necessary predictions based on recent used instructions. [75][76] related to on-chip caching for efficient cache utilization, [77] [78] based on data access frequency.

Following section explains the basic data access optimization techniques engineered to improve the cache efficiency.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

3.3.1. Data Access Optimization

Data access optimizations are generally code transformations with prime motivation, is to increase temporal locality by reordering iterations in a nested loop. These optimization techniques also used to expose parallelism and helps in vectorizing loop iterations. Compilers used heuristics to decide the effectiveness of applying these transformations. If nested loops are not perfectly nested then loop skewing, unrolling and peeling are used. Detailed information about these techniques can be found in [79][80].

Loop Skewing: When the carried dependencies prevents parallelizing, one can skew the loop-nest by modifying the aligning of iteration space coordinates and relabel the statement instances in new coordinates [81].

Loop Peeling: Unfolding few iterations of loop to eliminate loop-carried dependencies is known as loop peeling.

Loop Interchange: When order of loop is not important then loop-interchange transformation reverse the order of two adjacent loops in the loop nest making sure all dependencies are preserved. Loop interchange is used to improve locality, enhance parallelism, register reuse and vectorization [82].

Loop Fusion/Jamming: This technique involves fusion of two adjacent loops having same iteration space traversal into a single loop resulted in increased instruction level parallelism and data locality. The opposite of loop jamming is loop distribution, which divides the single loop in multiple loops with no carried dependencies. [82].

Loop Tiling: This loop transformation technique improves the data locality by increasing reuse of data in cache by increasing the depth of the loop nest [83]. Selection of best loop tile shape and size is a fundamental problem. Most of the current multicore processors have shared last level cache LLC and its space allocation depends on co-execution of applications, which may cause interference in shared cache [84]. Bao et al. [84] proposed a static compiler based defensive tiling to choose the best tiling size for optimal performance. Some of the work related to loop tiling includes [85][86] for reduction of capacity misses, [87][88][89] for reducing communication and [90][91][92] for auto/dynamic tunning of loop tiling.

3.3.2. Data Layout Optimizations

Data access optimization techniques may not be an optimal choice for data locality for computations with conflict misses while data layout optimizations improve spatial locality by arranging data structure and variables in memory. Some of the most commonly used techniques for data locality optimizations are inter and intra-array padding, array merging, array transpose etc. [82]. Parallelism and efficient exploitation of data locality have been considered as separate objectives in the literature. Kennedy et al. [93] explored this trade-off between data locality and parallelization by proposing memory model to determine cache lines reuse. The model uses loop optimization algorithms for efficient exploitation of data locality and in memory optimizations.

3.3.3. Cache Bypassing

The depth of cache hierarchies and cache size has been increasing to meet the high processing demands along with mounting more cores on chip. The performance of an application with little data reuse is severely affected by cache use. Researchers over the years engineered different techniques to effectively bypass cache to improve the performance of an application. The potential benefits of cache bypassing are obvious but bring many challenges including implementation overhead, memory and performance overhead etc. The details about different cache techniques, their potential benefits, challenges, and taxonomy is explained in detail by Sparsh Mittal [94].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

3.4. Locality-aware scheduling and Load-balancing

Applications need to exploit parallelism at multiple scales at fine granularity and across variety of irregular program and data structures and program inputs. Parallel algorithms demand extra space to enable temporal decoupling necessary for achieving parallelism as compared to sequential algorithms which attempt to minimise space usage. As applications are becoming more and more data intensive and task execution involves processing of huge volume of data. Optimal load balancing and locality aware scheduling is a critical issue to be resolved at highest priority for exascale systems. Locality optimization (horizontal - between processing elements and vertical- between levels of hierarchy)) has a direct impact on energy consumption for exascale systems. Scheduling millions of tasks per second within latency constraints is one of the major challenge and current centralized scheduling systems (Falkon [95], SLURM [96], SGE [97], Condor [98]) are deprived of handling fast scheduling. This issue is addressed by Ousterhout et al. [99] by presenting stateless distributed scheduler and called it sparrow which supports data-aware scheduling to help collocating task and input data.

Load balancing can be achieved by work stealing but random migration of tasks resulted in poor data locality. Load balancing is a challenging task in fully distributed environment as scheduler has information about its own state. Load balancing have been extensively studied over the years and one can broadly classify load-balancing techniques into static and dynamic techniques. These techniques achieved optimal load balancing in centralized or distributed manner. Although work stealing is widely used efficient load balancing technique e.g. OpenMP [100], Cilk [101], X10 [71], but random work stealing results in poor scalability on large scale systems [102] . Falt et al. [103] proposed a locality aware task scheduling for parallel data stream systems.

Muddukrishna et al. [104] proposed a locality aware task scheduling and runtime system assisted data distribution algorithm for OpenMP tasks on NUMA systems and many-core processors. Ding et al. [105] proposed a cache hierarchy aware loop–iterations-to- core mapping strategy by exploiting data reuse and minimising data dependencies which results in improved data locality. Lifflander et al. [106] proposed locality aware optimization at different phases of fork/join programs with optimal load balance based on Cilk and also provides programmatic support for work stealing schedules which helps in user guidance on data locality.

Xue et al. [107] proposed a hybrid locality aware dynamic load balancing and locality aware loop distribution strategy to multiprocessors and enhanced performance is reported compared to other static/dynamic scheduling techniques. Isard et al. [108] proposed multipurpose execution engine and called it Dryad for coarse-grain data parallel applications for efficient fault resilience, data transportation and data aware tasks scheduling. Maglalang et al. [109] proposed locality aware dynamic task graph scheduler with optimal locality, load balance and minimum overhead.

Yoo et al[110] proposed a locality aware task scheduling for unstructured parallelism by developing a locality analysis framework. The offline scheduler takes workload profuse information as input and makes scheduling decisions that are optimised with underlying cache hierarchies. Paidel et al. [111] work focused on the selection of tasks that are most favourable to migrate across nodes in distributed environment which is further supported by application level data locality. Choi et al. [112] proposed locality aware resource management and workflow scheduling by balancing resource utilization and achieved data locality based on network bandwidth in HPC cloud environment.

Work scheduling and stealing are two most commonly used scheduling paradigms for scheduling multithreaded computations to workers in a typical task based parallel systems. Guo Yi [113] in his PhD work proposed a locality aware work stealing framework for efficient exploitation of data locality (affinity) and adaptive work-stealing scheduling algorithm.

 doi:10.20944/preprints202211.0161.v1

Hindman et al. [114] proposed Mesos, a thin resource-sharing layer with prime objective to engineer a scalable and efficient system for sharing resources between heterogeneous frameworks by presenting an abstraction called a resource offer. The resources offered to framework are decided by Mesos based on organizational policy, while which policies to accept, and tasks to run on them are decided by framework. Mesos use delay scheduling and Data locality is achieved by taking turns to reading data stored on each node. Isard et al. [115] proposed a fair scheduling of concurrent jobs with fine grain resources sharing for distributed computing clusters and name it Quincy which achieved better fairness and improved data locality.

3.5. Bulk Synchronous Processing (BSP)

The BSP model was presented by Valiant [116] and modified by McColl, is a bridging model to design parallel algorithms and mainly consist of processing components, equipped with local memory, a network for communication and synchronization between components. Communications is facilitated by one-sided put and get calls rather than two way send and receive operations. Barrier ensures that all one-sided communication is completed. The oversubscription of processing elements and problem decomposition are exploited by BSP model for automatic distributed memory management. Logical processes are randomly assigned to processors for optimal load balancing and communication [117]. Google used BSP for graph analytics with Pregel [118] and map-reduce while some open source projects (Apache Hama [119] and Giraph [120]) also extended the use of BSP by employing high performance parallel programming models on top of Hadoop. There are different programming languages and interfaces based on BSP model including BSPLib [121], BSPonMPI [122], Bulk Synchronous Parallel ML (BSML) , Multicore-BSP [123][124].

3.6. Out-of-Core Computing

Out of core algorithms (e.g. solvers for large systems of linear equations as in nuclear physics) are used when data to be processed is too large to fit in memory and data needs to be fetched from storage devices e.g. hard and tape drives or from memory attached via network. As these auxiliary storage devices are slow and acceptable performance is achieved by data re-use in memory and how data is laid out in these storage devices for I/O on larger blocks [125].

The use of virtual memory which enables programmers to access much larger then available memory without need to know the actual location of data in memory hierarchy. Different techniques proposed over the years to efficiently exploit the data movement across different memory hierarchies i.e. caching, swapping and demand paging etc. There are applications where virtual memory systems do not meet the programmer's expectations and does not provide enough virtual memory space. Out of core algorithms are used when primary and virtual memory are not enough to hold application data [126]. The principle of locality plays an important role in performance of an application and locality in terms of out of core algorithm means that data must be laid out in storage device to allow blocks of data exchange between storage devices-memory and also re-use of data in memory. In a traditional disk-based approach, processor remains idle until the data is loaded in to memory and next read is not initiated until the computation is finished. The author in [127] addresses this issue by proposing a two process approach where disk I/O and computation is performed concurrently. To overcome the limitations of speed at which data can be accessed from storage devices, is use of shared and distributed memories across the cluster, which demands use of high speed inter-connects (Infiniband) for dataset to be loaded before the start of algorithm in large aggregated distributed/shared memory. The use of high speed inter-connects though improve the performance but on the expense of tangible cost of initial setup, maintenance and energy consumption over time. The trend of use of non-volatile memory NVM e.g. low power flash-based Solid-State Drives (SSDs) to speed up the I/O is increased over the years. These SSDs are used

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

along with traditional storage devices on I/O nodes in cluster environment, which results in enhanced performance with faster data access/load from these SSDs to I/O nodes [128].

Jung et al. [128] addresses the issues and potential benefits of co-locating the non-volatile memory and compute nodes by presenting an compute local NVM architecture. They identify the drawbacks of modern file systems and proposed a novel Unified File System (UFS). Besides there are numerous efforts in the literature focused on usage of SSDs that include use of SSDs as caches [129] FlashTier [130], Mercury [131], Salue et al. [132] presented an out-of-core task-based middleware for data intensive computing and [133][134][125] is related to out of core algorithms for solving dense and linear systems of linear equation.

3.7. Parallelism mapping

The increase in system concurrency introduced a massive challenge for application and system software to deal with large scale parallelism. The widely deployed message passing model MPI may not be a suitable choice to deal with demands of extreme scale parallelism for exascale systems. Finding a single anomalous process among millions of running processes and threads is not an easy task [135]. The issues related to parallelism are diverse and are very much program dependent. The parallelism mapping is very much platform/hardware dependent and optimal mapping decisions depends on many factors like scalability (how much potential parallelism should be exploited), number of processors in use, scheduling policies, relative cost of communication and computation etc. There have been various efforts to address this issue. In multi-cluster environment, the bandwidth among nodes inside a single cluster is normally much higher than the bandwidth between two clusters similarly within node cores sharing cache can communicate much faster. Entities ex- changing or sharing lots of data could be placed on hardware processing units physically close to each other. By doing so, the communication costs are reduced, thus decreasing the application's overall execution time and as a consequence its energy consumption [66]. Along with communication, application performance is also dependent on Load imbalance, communication pattern and memory usage.

Mapping threads/processes to cores is depending on many factors like operating system, implementation (Different implementations of MPI e.g. OpenMPI, MPICH, IntelMPI) and runtime system i.e. Message Passing Interface (MPI), Partitioned Global Address Space (PGAS). The work related to efficient mapping of processes to reduce the communication cost is based on finding the communication topology (communication pattern/graph of the application e.g. number and size of messages between processes etc.) and network topology graph (e.g. the latency and bandwidth between different processor cores, inter and intra node communication cost etc.) and then appropriate selection of optimal cores where the processes should be mapped. One can achieve the task of optimal mapping of processes to cores by running an application with monitoring tools to understand the communication pattern. Trace libraries can provide the communication details e.g. MPI Trace [136]. There is lack of standardization for thread/process mapping at start-up but can be implemented at MPI execution level. Unfortunately, current parallel programming paradigms seems not viable to address the data locality issue to improve parallel applications scalability. There is a need for some evolutionary and revolutionary changes in parallel programing models to address these problems.

There is considerable amount of work in literature related to process placement for MPI application based on correlating the communication and network topology by algorithmic means using graph theory and implementation based on scheduler, compiler or exploiting MPI runtime environment.

3.7.1. Message Passing interface support for process mapping

The HPC application community has begun experimenting with manual placement of the individual processes in a parallel job commonly referred to as "process placement" or "process affinity. MPI implementation provides different mapping patterns like by-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

node (a.k.a., scatter, cyclic) and by-slot (a.k.a., bunch, pack, block). The different MPI implementations also provide numerous mpirun command line options bind with different runtime configuration parameters to enhance the process mapping and binding. Assigning more than one process to a single processor is considered as oversubscribing in most HPC environments and is generally discouraged as MPI/ HPC applications are CPU-intensive; sharing multiple processes on a single processor cause starvation and performance degradation [137].

Given a parallel application, it is essential to efficiently map the MPI processes to the processors on the nodes. Communication pattern needs to be understood by running application with the profiling tools. Trace libraries can provide the communication details e.g. MPI Trace and process mapping can be done manually. Communication assisted communication analysis can also be performed to map MPI processes to processors. The ultimate goal is to reduce communication by mapping processes with frequent communication on a single node. There could be a number of taxonomies to classify the mapping methodologies, like target architecture based, optimization criteria based, workload based (Static or dynamic) etc. Static mapping methodologies are best suited for static workload scenarios where a predefined set of applications with known computation and communication behaviour and a static platform are considered. As optimization is performed at design-time, the methodologies can use more thorough system information to make decisions for both homogeneous and heterogeneous architectures [42].

3.7.2. Algorithmic Approaches for process mapping

The speed of communication among cores in a multi-core processor chip (intra-chip) varies with core selection, since some cores in a processor chip share certain levels of cache and others do not. Consequently, intra-chip inter-process communication can be faster if the processes are running in cores with shared caches than otherwise. And situation gets even worse when among cores on distinct processor chips in a cluster. Rodrigues et al. [138] used a graph mapping technique for mapping process to cores by considering intra-chip, intra-node and inter-node communication cost to improve the performance of applications with stable communication pattern. The approach was tested by comparing execution times of a real-world weather forecast model, using the default mapping and proposed solution and obtained an improvement of up to 9.16%.

Rashti et al. [139] merged the node physical topology with network architecture and used graph embedding tools with MPI library to override the trivial implementation of the topology functions and effectively reorder the initial process mapping. Hestness et al. [140] presented the detailed analysis of memory system behavior and effects for applications mapped to both CPU and GPU cores. Understanding the memory system behavior is very important as multiple cores are integrated on the same die that share numerous resources. This paper presents a detailed comparison of memory access behavior for parallel applications executing on each core type in tightly controlled heterogeneous CPU-GPU processor simulation.

Communication topology of the application and underlying architecture affect the performance of point-to-point communication and MPI provides different primitives to gather such information like MPI Cart create and MPI Graph create. Application communication graph can be created by calculating the communication cost between processes using message count and message volume. HU chen et al. [141] propose a profile-guided approach to find the optimized mapping automatically to minimize the cost of point-to-point communications for arbitrary message passing applications and name it MPIPP (MPI process placement toolset). This tool gets the communication profile of the MPI application and also gets the network topology of the target clusters. They also proposed an algorithm for optimized mapping and enhanced performance is reported by comparing their solution with the existing graph portioning algorithms.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

Zhang et al. [142] proposed an approach for optimised process placement to handle collective communication by transforming them into series of point-to-point communication operations. They decomposed a collective communication in to point-to-point and then generated the communication pattern of the whole application. They used graph-partitioning algorithm for optimised process mapping. Pilla et al. [143] proposed a topology aware load balancing algorithm for multicore systems by modelling a distances and communication among hardware components in-terms of latency and bandwidth by exploiting the properties of the current parallel systems i.e. network interconnection, multi-levels of cache etc. The introduction of multi-core processors introduces numerous challenges including contention of various physical cores for using shared resources. With more cores in a single node causes multiple requests for the network interface that results in performance degradation [144]. This demands the distribution of parallel processes in available computing nodes such that requests arriving to each network interface be decreased, queuing time of messages at interface queues will be decreased as well thus resulting in enhanced performance. Zarrinchain et al. [144] addressed this issue by proposing a solution for mapping parallel processes to multi-core clusters so as to reduce network interface contention by determining length of the messages among processes and an appropriate value for the threshold (number of processes in each compute node) using the number of adjacent processes of each process and the number of available free cores in the computing nodes.

Guillaume et al. [145] proposed an efficient process mapping of MPI application to better take advantage of a multicore environment without any modification of MPI implementation and improved performance is reported solely on the basis of relevant process placement. They extract an embedding of the application's graph from the target machine's graph and used scotch software to solve NP graph problems. Scotch applies graph theory, with a divide and conquers approach, to scientific computing problems such as graph and mesh partitioning, static mapping, and process ordering. The information is then used to create a mapping between MPI process ranks and each node's core numbers. Finally, an application-specific command line is generated.

Other work related to mapping includes architecture specific mapping [146][147][148] for Blue Gene systems, [139][149][150] targeting multicore networks, [151] targets hybrid MPI/OpenMP mapping, [152] is mapping library,[153] [154] are programming standards that support virtual topology mapping.

3.7.3. Machine learning based parallelism Mapping

Programming with target architecture in mind and mapping parallelism to processors/cores to avoid/minimize communication are two alternative ways of optimizing application performance. Selecting the correct mapping scheme has significant impact on the performance and these mapping schemes are very much architecture dependent. So there is always a need for an automatic and portable solution for assigning tasks to target architecture to achieve scalable parallelism.

Castro et al. [155] proposed a machine learning-based approach to do efficient thread mapping in transactional memory (TM) applications. Software TM libraries usually implement different mechanisms to detect and solve conflicts. As a consequence, it becomes much more complex to determine a suitable thread mapping strategy for an application since it can behave differently according to the conflict detection and resolution mechanisms. Grewe et al. [156] proposed a portable partitioning scheme for OpenCL programs on heterogeneous CPU and GPU architectures by extracting code features statically and using ML algorithms for predicting best task partitioning. Tournavitis et al. [157] proposed profile driven parallel detection and ML based mapping to overcome the limitations of static analysis and traditional parallelizing compilers by using profiling data to extract control and data dependencies and then used ML based trained predictor to select a best scheduling policy offered by OpenMP i.e. CYCLIC, GUIDED, STATIC AND DYNAMIC.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

Wang et al. [158] proposed a compiler based automatic and portable approach for selecting best thread scheduling policy based on ML model learnt offline to map parallel programs to multi cores. ML-based predictors use profiling information to Characterize code, data and runtime features of a given program. The feature extractor needs several profiling runs for a program to extract these features. They used artificial neural Network ANN and Support Vector Machine SVM to build two different learning models to predict program scalability and to classify scheduling policies. The models were trained using a set of training data that consisted of pre-parallelized programs with selected features and desired mapping decisions. Long et al. [159] used a machine Learning based approach for the cost aware parallel workload allocation by using static program features. More specifically they used ML to determine the thread number allocated for a parallel java loop on the run time and doesn't tackle portability. Adaptive multi-versioning for OpenMP parallelization via machine learning integrated in the compiler to achieve parallelism. Pinel et al. [160] proposed a machine learning based automatic parallelisation of a scheduling heuristic, Savant, by defining a generic parallel pattern-matching engine that learns the algorithm to parallelize.

Emani et al. [161] proposed a predictive modelling approach that dynamically considers a number of thread selection policies and choose the one it believes will perform best at every parallel loop and called this approach as mixture of Experts. Their approach predicts the optimal number of threads for a program and also predicts the run time environment. Emani et al. [162] proposed an efficient parallel mapping based on online change detection by combining offline model with online adaption to find the optimal number of threads for an OpenMP program. Luk et al. [163] proposed a fully automatic mapping technique to map computations to processing elements on heterogeneous multiprocessors. They measured the parallelization speedups on matrix multiplication with heterogeneous machine and implemented it in an experimental system called Qilin and report the performance close to manual mapping with adaptability feature for different problem sized and hardware configurations.

Dominguez et al. [164] extended their previous work by using Servet to map applications on multicore systems and analyse the performance on different parallel programming models i.e. message-passing, shared memory and Portioned Global Address Space (PGAS). The featured extracted by Servet can be used to optimised the performance by choosing more appropriate mapping policy without source code modification.

3.8. In-Situ Data Analysis

The increasing data size, limited storage and bandwidth, efficient use of compute resources, difficulties in examining output data, storing and retrieving output data for post data analysis is considered impractical for Exascale environment. The term data movement for in situ data analysis is mostly used to describe the data movement back and forth to persistent storage and retrieving data for post data analysis. In situ analysis translates to saving in execution times, power, storage cost and avoiding either completely or to very large extent massive data movement of simulations output to persistent storage.

In-situ data analysis is performed on the data in the transition phase before it is written back in to the parallel file system. Tiwari et al. [165] exploit the compute power in Solid State Drives SSDs for in situ data analysis and called it Active Flash. This approach provides energy efficient data analysis as computation near storage devices reduces the data movement cost and also SSDs are equipped with low power, multi-core ARM based controllers. In situ analysis has become one of core aspect of data interpretation in big scale scientific simulations. But for data that is already resided in backend storage system, efficient data analysis is still a core issue.

Zheng et al. [166] proposed a in situ middleware system to facilitate the underlying scheduling tasks e.g. cycle stealing. They created an agile run time and called it GoldRush, a fine-grained scheduling to steal the idle resources by ensuring the minimal interruption to the simulations and in situ data analysis. The system makes the use of idle

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

wasted resources of compute nodes to be efficiently used for in situ analysis. The experiment results showed enhanced performance, low data movement cost, efficient resource utilization and minimum interference with simulation. Sewell et al. [167] proposed a framework that uses both in-situ and co-scheduling approaches for large scale output data. They compare the performance by analysing different setups to perform data analysis i.e. in situ, co-scheduling and combination of both.

3.8.1. In-Situ compression

Scientific data is mostly regarded as effectively incompressible due to its inherently random nature and decompression also imposes extra overhead. Sriram et al. [168] addresses this problem of compression by exploiting temporal patterns in scientific data to compress data with minimal overhead on simulation runtime.

Zou et al. [169] worked on data compression for the removal of redundant data reduction by using general compression techniques and the proposed a use-specific methods that allows users to remove redundant or non-critical data by using simple data queries. This method allows users to optimise output data and explicitly identify the data that needs to be retained. The general-purpose lossy compression techniques do not provide this level of flexibility.

3.8.2. Use of indexing for in situ data Analysis

As the computation power increases at the brisk speed compared to storing data to disks and reading data back from these disks. J Kim et al. [170] used indexing and in situ processing to address these challenges. Indexing is a powerful and effective way of addressing the data access issue while the implementation of indexing technology is embedded in DBMS which lack the ability to manage most scientific data sets. They used in situ data processing for creating indexing in parallel thus reducing the resources utilized to store data back and forth from disks. Usage of indexes improved the data access time and in situ data processing reduced the index creation time.

According to Sriram et al. [171] current state of art indexes requires compute and memory intensive processing thus make indexing impractical for in situ processing. They propose DIRAQ, a parallel in situ, a query driven visualization and analysis during simulation time that transform the simulation output to a query accessible form. This technique has a minimum overhead on simulation run and speed up query response time.

Yu Su et al. [172] proposed in situ Bitmaps generation and performing data analysis based on these Bitmaps. Bitmap indices can be used as a summary structure for offline analysis tasks. Their work basically focused on in situ analysis of selected bitmaps thus reducing the amount of simulations output data to be stored on disk and reducing the memory requirements for in situ analysis. HPC systems with in situ data analysis analyse temporary data sets as they are generated. Permanent data sets that are stored in the backend persistent storage systems; their efficient analysis is still a challenging task.

3.8.3. In situ Visualization

As scientific simulations produce huge volume of raw data and saving vast amount of raw data for off line analysis is a complex task and not a suitable method for current Peta scale and future Exascale systems. Karimabadi et al[173] addresses this I/O issue through in situ visualization strategies. The main idea is to extract important features from raw data parallel with simulation and thus reducing the amount of raw data stored on the disk. Their work focused on overhead associated with computation needs for in situ visualization in parallel with simulation run.

Yu et al. [174] investigated in situ visualization for turbulent-combustion simulations and explored in situ data processing and visualization strategies in an extreme parallel environment. Their results showed that in situ visualization enhanced the performance and can be used for accelerating high-performance supercomputing and scientific discov-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

ery. Zou et al. [175] proposed an online data query system (FlexQuery) using inline performance monitoring and minimize data movement with low latency data query execution. They demonstrated the dynamic deployment of queries by proposed query system for low latency remote data visualization.

Woodring et al. [176] addresses the issue of reducing the memory footprints by sharing data between visualization libraries and simulation by using zero copy data structure. They optimised the traditional way of coupling of different mesh-based codes for in situ data analysis where data needs to be explicitly copied from one implementation to another with necessary translation. This results in redundant data, which ultimately increase memory footprints. They proposed an alternative way of sharing data between simulations through optimised dynamic on demand data translation with reduced memory footprints and memory per core. Nouanesengsy et al. [177] proposed a generalized analysis framework for automatically evaluating the relative importance of data in order to produce reduce data products thus ensuring enough resources to process reduce data sets. The proposed framework prioritizes large-scale data by using user defined prioritization measurements.

3.8.4. In situ feature selection

Landge et al. [178] proposed in situ feature extraction techniques which allows state of art feature based analysis to be performed in situ with minimal overhead on simulation runtime.

Zhang et al. [179] proposed a framework for in situ feature based object tracking on distributed scientific data sets with decentralized online clustering DOC and cluster tracking algorithm. They run in parallel with simulation process and obtain data from on chip-shared memory directly from simulation. Their results showed that proposed framework can be efficiently utilized for in situ data analysis of large-scale simulations.

4. Data Locality in Big Data Environments

Big data has transformed our societies with many innovative applications [180]–[189]. Different solutions emerged over the years to deal with big data issues and successfully implemented. But never the less, these solutions do not satisfy the ever-growing demands of big data. The issues related to big data are immense and cover variety of challenges that needs a careful consideration. These challenges include data representation, redundancy reduction, data compression, data life cycle management, analytical mechanism, data confidentiality, energy management, expandability and scalability, high dimensionality, computational complexity, real and distributed computation and non-structured processing etc. The key advantage that big data technologies brought over traditional HPC is of data locality. Hadoop brings computation to data and Spark further enhance by in-memory computation.

Big data analysis is done at various levels i.e. when server memory is huge, in memory analysis can be used by keep the hot data in memory for efficiency. This memory level technology is idle for real time analysis e.g. MangoDB [1] Business intelligence BI has different tools and procedures for analysing data when it exceeds the memory capacity. Map/reduce is used most widely for massive data analysis, which is beyond BI capacity and is mostly fall into off line analysis category [190]. Chasing a right solution depends on the size of data, urgency of results, prediction about the needs of more processing power as the size of data increases, fault tolerance for applications in case of hardware failure, data rate and scalability etc. Data locality has been recognized as one of the major issues to be resolved for exascale systems and considerable efforts have been done to move computation closer to data. Map-Reduce [191] is most widely used data processing model in data centric computation environment. Google map-reduce based on Google File System (GFS) [192] achieves locality by relying on data aware task scheduling and block replication. Hadoop [193] based on HDFS (Hadoop Distributed file System) [194] followed the same approach. Different solutions emerged as the processing requirements of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

application changes i.e. HBase [195] for random access data, Apache Giraph [120] for graph processing, Spark streaming, TEZ [196] Twister [197] for iterative streaming. Data locality in data parallel systems by bringing flexibility to scheduling algorithms has a major edge over traditional HPC systems. In the following section we analyze the research efforts and current trends by focusing specifically on enforcing data locality in big data environment.

4.1. Parallel Programming Models

We can broadly classify parallel programming models in big data environment as Batch processing and iterative. Following section gives the brief over view of both.

4.1.1. Batch Processing

Map-reduce is probably the most widely used data centric programming model where a piece of work is divided among several parallel map/reduce tasks. The original implementation by Google relies on Google File System GFS [192], whereas open implementation of map-reduce (Hadoop [193]) relies on Hadoop Distributed File System HDFS and data locality is achieved by data aware scheduling and block replication. The input files are split up in input format, which select the file, defines the inputs splits, breaks the file into tasks and provides the place for the record reader objects. The input format defines the list of tasks that makes up the map phase. The task then assigns to the node of the system based on where the input file chunks are physically resident [198] . The input split describes the unit of work that comprises a single map task in a map-reduce program. The record reader loads that data and converts it into (key value) KV pairs that can be read by the mapper. The mapper performs the first phase of the map-reduce program given the key and the value, the mappers export key and value pairs and send these values to the reducers. The process of moving map outputs to the reducers is known as shuffling [199].

Due to the lack of support for processing multiple heterogeneous datasets, map-reduce-merge was proposed by Yang et al. [200] to facilitate the use of map-reduce in relational operations like join.

4.1.2. Iterative

Traditional map-reduce lacks the support for the iterative tasks. Mapper needs to read the data and after each iteration results need to be written back to disk for subsequent iteration. Disk I/O is a bottleneck here and for each iteration new mapper and reducer needs to be initialized. [201]. Different wrappers or extensions have been developed to overcome the shortcomings of Hadoop for improved performance e.g. programming model extensions.

Map-iterative-reduce [202] is an iterative framework emerged from map-reduce (which lacks the support from reduce-intensive workloads) to support reduce-intensive applications. One of the implementations of iterative-map-reduce is Twister. Ekanayake et al. [197] presented a programming model and architecture of Twister and compared its performance with other programming models like Hadoop and DryadLINQ and efficient iterative map-reduce computation performance is reported. Bu et al. [203] also targeted the lack of support of Map-reduce and Dryad for iterative programs by proposing Haloop for efficient iterative data processing. They further enhanced the performance by presenting loop aware scheduling and efficiently exploiting various cache mechanisms.

Spark [204] provides data flow execution engine that supports cyclic data flows, with support of various langauges e.g. Scala, Python, Java. Sparks abstracts data with Resilient Distributed Datasets (RDD), which is spark representation of set of data, distributed across multiple machines and allows fault tolerant in-memory computations on large cluster.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

4.1.3. Language Support

Map-reduce programming model often refer as a low-level model for analysts, who are used to of SQL like or declarative languages. It also requires advance programming skills and in-depth understanding of system architecture for developing efficient map-reduce applications. [205]

Hive provides the ability to analyze large amounts of data stored in HDFS. Hive was designed to appeal to the community comfortable with the SQL and uses SQL like language known as HiveQL. It supports map and reduce transforms scripts in the language of the user choice, which can be embedded with the SQL, and is widely used in Facebook. Hive is a framework for performing analytical queries while its dominant use is to query flat files. Currently Hive can be used to query data stored in Hbase. The worker nodes in Hive keep small tables in distributed memory setup for quick data access.

Olston et al. [206] developed programming language (Pig-Latin), which gives programming abstraction of Java map-reduce (low level procedural style) to make it somewhat similar to high-level declarative style of SQL for RDBMs.

4.1.4. Locality aware partitioning

Map-reduce based join operation is not optimised when dealing with skewed data and proposed solutions often result in huge volume of data in the shuffle phase affecting the performance of map-reduce based join. Lin et al. [207] addresses this problem by proposing SALA (A Skew-Avoiding and Locality Aware) algorithm and locality aware partitioning to ensure data locality, even data distribution to reducers, without any modification of map-reduce framework. On the similar line, Ibrahim et al. [208] proposed LEEN for locality aware and fairness-aware key partitioning by incorporating asynchronous map and reduce scheme. Their results show enhanced data locality with minimum intermediate shuffle data. Rhine et al. [209] proposed locality aware scheduling algorithm and input data split, by partitioning data belonging to a node, in single split.

4.2. Data Placement

4.2.1. Locality aware Data Placement

Hadoop relies on HDFS to store data with high availability to multiple nodes but data is placed randomly to achieve load balance without taking characteristics of data into consideration. By default, Hadoop lacks the ability to collocate data on the same set of nodes. Eltabakh et al. [210] addresses this issue by proposing CoHadoop, an extension of Hadoop, giving control to applications to manage locality aware storage by modifying HDFS data replacement policy. Minimizing off-switch communication increased the performance of map-reduce in Hadoop. Yu et al. [211] argues that grouping of blocks of data in fewer racks, enhanced the performance by reducing off-chip communication and proposed the methodology to group data and scheduling mechanism by exploring trade-off between off-chip communication and parallelism.

Schedulers often consider map tasks for locality and ignore reduce tasks while fetching intermediate data which resulted in performance degradation. Tan et al. [212] proposed the stochastic optimization framework for improving reduce task data locality for sequential map-reduce jobs. Wang et al. [213] addresses the issues related to random distributed of data placement in traditional Hadoop by proposing data grouping aware (DRAW) data placement scheme. DRAW optimizes group sizes and optimize parallelization per group by re-organizing data layouts.

4.2.2. Locality aware data placement in heterogeneous environment

The Hadoop implementation by default assumes that cluster nodes are homogeneous and data locality is not taking into consideration to map tasks. Xie et al. [214] addressed this issue by proposing a data placement strategy in heterogeneous environment (Hadoop Cluster) for optimal load balance. The data placement algorithm must consider the node heterogeneity (processing capabilities) to partition input and intermediate data.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

Arasanal et al. [215] proposed load balanced data placement enhancements and input data distribution algorithm in Hadoop based on processing capabilities of the nodes. The similar approach proposed by Wei Lee [216] at al by proposing a dynamic data placement algorithm to balance workload based on the computing capabilities of each node in heterogeneous environment. The proposed strategy dynamically adjusts workload and reduces the data transfer time.

Ubarhande et al. [217] analyzed various scheduling techniques from data locality prospective and proposed data placement methodology based on computation ratio for Hadoop data nodes and enhanced performance is reported by executing standard map-reduce applications i.e. Grep and word count. The input to the task must be present on a node where task is supposed to be executed otherwise needs transferring input data which ultimately increased execution time. Sujitha et al. [218] proposed a methodology to address the issues of heterogeneity and data locality in Hadoop.

4.3. Scheduling and Load Balancing

4.3.1. Locality-aware Scheduling and Load Balancing

Locality aware scheduling is one of the prominent features of MapReduce, which allows schedulers to bring compute to data rather than vice versa. The cross-rack traffic must be reduced for optimal performance. Guo et al. [219] proposed an algorithm to improve data locality by exploiting available resources and considering all tasks together rather than task-by-task approach as in traditional Hadoop. They also proposed another algorithm by integrating fairness and locality by allowing users to define trade-off for desired performance. Chen et al. [220] proposed a partition algorithm CLP (Cluster Locality Partition) for optimal load balance and performance. Locality partitioning achieved data locality by assigning data clusters to appropriate nodes.

Network traffic is a major bottleneck in data intensive applications and can be reduced by locality aware scheduling. Chen et al. [221] addresses this issue and proposed LaSA locality aware scheduling algorithm for data aware resource assignment. Wang et al. [102] proposed a data aware work stealing technique implemented at node /scheduler level to achieve enhanced load balance and efficient exploitation of data locality by proposing a fully distributed task scheduling system for Many Task Computing (MTC) systems. Park et al. [222] proposed runtime reconfiguration scheme (Dynamic Resource Reconfiguration DRR) which schedules task to nodes where data resides and also dynamically increasing or decreasing the computing capability of each node for optimal data aware scheduling. Zaharia et al. [223] addresses the conflict between data locality and fairness in scheduling by proposing delay-scheduling algorithm, which increases throughput with optimal data locality and guaranteed fairness. Hadoop has been optimized to reduce the amount of network traffic i.e. delay scheduling achieves nearly optimal data locality for variety of workloads which ultimately result in low volume of network traffic [223]. Intermediate data shuffling in Hadoop still generate huge volume of network traffic.

4.3.2. Locality aware scheduling and load balancing in heterogeneous environment

Scheduling of map-reduce tasks is further complicated in heterogeneous environment, addressed by Zhang et al. [224] by proposing locality aware scheduling algorithm. Hsu et al. [225] proposed locality and load aware virtual machine mapping techniques to improve map-reduce performance in heterogeneous environment by portioning data before map phase and virtual machine mapping in reduce phase. Xue et al. [226] proposed dynamic scheduling algorithm BOLAS (Bipartite-Graph Oriented Locality Aware Scheduling) by modeling scheduling problem as bipartite-graph matching problem using Kuhn-Munkeres algorithm and achieved improved data locality for both homogeneous and heterogeneous environment. Sadasivam et al. [227] proposed Hybrid Particle Swarm Optimization-Genetic Algorithm (HPSO-GA) for efficient execution of tasks and utiliza-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

tion of resources achieved by capacity aware load distribution in heterogeneous environment. Zhang et al. [228] addressed the issues related to performance of map-reduce in heterogeneous environment by proposing a methodology to separate map shuffle and reduce stage for optimized task allocation and controlled dynamic execution.

4.3.3. Adoptive Scheduling

Guo et al. [229] proposed a data distribution aware task scheduling methodology by overcoming the uneven data distribution strategy of default scheduling techniques. The network overhead is reduced and high system efficiency is reported by mapping tasks to nodes with high probability of data availability and task's scheduling priority. Hammoud et al. [230] proposed locality aware reduced task scheduler and name it LARTS which co-locate data with reduce tasks. LARTS achieves high data locality and reduce scheduling delay/skew and exploit resources efficiently. Ahmad et al. [231] proposed communication aware load balancing and scheduling of map computation, predictive load balancing for reduce computation to optimize the performance of map-reduce in heterogeneous environment. On the similar line Kumar et al. [232] proposed context aware scheduling technique (CASH), Zhao et al. [233] proposed job scheduling algorithm on map-reduce, Hammoud et al. [234] presented locality and skew aware scheduling algorithm (CoGRS) and Ibrahim et al. [235] proposed replica aware scheduling for map-reduce. Dtailed survey on adoptive scheduling in map-reduce is presented by Mozakka et al. [37].

4.3.4. Delay Scheduling

Delay scheduling involves scanning jobs more than once before some threshold is reached after which job is scheduled. Delay scheduling may also delay high priority jobs. Sethi et al. [236] proposed mechanism to force scheduler to launch high priority jobs to be executed locally or on some nodes based on the availability of data. Yang et al. [237] proposed a scheduling algorithm by segregating map and reduce as separate stages of scheduling problem and addresses the issues of map-reduce stage deadline, execution time and data locality. Bezerra et al. [238] proposed data locality aware job scheduling with prime motive is to run tasks which handle the same blocks of data on the same node where blocks are residing.

4.4. In-Memory Computations

The memory storage capacity and bandwidth is increasing at brisk speed and time is not far when memory will replace hard drives. This transformation will ultimately set trends for building in-memory systems where major portion of data fits in memory for the obvious performance gain. In-memory capacity of machines in map-reduce clusters is often underutilized and can be effectively utilized by in-memory prefetching input data to improve data locality. Sun et al. [239] proposed a prefetching service based task scheduler HPSO (High Performance Scheduling Optimizer) to predict the optimal node for future tasks and prefetching needed data.

Both Hadoop and Spark [240] are big data frameworks and do perform the same tasks, are not mutually exclusive and able to work together. Spark is reported to work 100 times faster than Hadoop in some situations and does not have its distributed storage system [241]. Apache Spark is most widely used distributed in-memory computing framework that handles in memory operations by copying data from distributed file systems in to faster logical RAM [242]. Map-reduce writes all data back to distributed storage system after each iteration to ensure full recovery whereas Spark arranges data in resilient distributed datasets that are capable of full recovery in case of failure. The efficiency of Spark is questioned for applications where dataset is loaded and evicted at runtime. Shen Li et al. [243] addresses this problem by proposing Stark for in optimizing in-memory processing on dynamic dataset collection by avoiding replication and shuffling. Engle et al. [244] proposed Shark (Hive on Spark) for deep data analysis based on Resilient Distrib-

 doi:10.20944/preprints202211.0161.v1

uted Datasets (RDDs) [204] to achieve scalability, performance, resilience and efficient execution of iterative algorithms with intra-query temporal locality for in-memory computation on large clusters. Sentos-Neto et al. [245] exploited storage affinity by data reuse and used replication strategy for efficient scheduling without any runtime information. Reynold et al. [246] proposed GraphX by combining data parallel and graph parallel systems by efficiently distributing graphs, exploiting resilience and in-memory computation.

4.4.1. Registers and Cache centric optimizations

The effective and efficient usage of registers is usually targeted at compiler or assembly language level. The in-memory data bases with traditional iterative-style queries often results in poor data locality [33]. Efficient utilization of different levels cache hierarchies is of paramount importance for the obvious performance gains. The work related to cache optimization includes compression [247], colouring [248], re-organizing data layouts by organizing records in columns lay out [249][250] or using decomposition storage model [251] and cache conscious indexes [252][253]. A detailed survey is provided by Zhang et al. [33].

4.4.2. Non-Uniform Memory Access (NUMA)

Each processor in NUMA architecture has faster access to its own local memory and relatively slow and higher latency access to remote memory. In the context of NUMA, much of the work in literature is related to data partitioning [254][255][256] and data shuffling [257]. A detailed survey is provided by Zhang et al. [33].

4.4.3. NVRAM

The use of non-volatile memory is an emerging trend in high performance computing environment to provide memory with high speed and capacity. The NVRAM technology provides better performance compared to traditional hard drives/flash drives and comparable performance to DRAM e.g. Phase change memory technology [258] Memristive devices [259] STT-MRAM[260]. A detailed survey is provided by Zhang et al. [33].

4.4.4. In- Memory Data Processing Systems

In- memory data processing is much faster comparing to other data centric computational models e.g. Hadoop and is often used for the analysis of huge volume of data within time constraint. Spark [240], Mammoth [261], Piccolo [262] are data analytics systems where as S4 [263], Map-reduce online [264] are real time data processing systems.

4.4.5. In-Memory data storage systems

In-memory data storage systems include relational, NoSQL databases and cache-based system (cache between application server and data base). [265][266][267][268] are few examples of in-memory relational databases, [269][270][271] are in-memory NoSQL data bases, [272] [273][274] are in-memory cache-based systems. A detailed survey is provided by Zhang et al. [33].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

Table 1. Data Locality aware research efforts in HPC & Big data Environment and Convergence Challenges.

	HPC	Big Data	Convergence Challenges
Parallel Programming Models	Majo et al. [67], Lezos et al. [68], Regan-Kelley et al. [69], X10 [71], Huang et al. [72], BSP [116], Pregel [118], including BSPLib [121], BSPonMPI [122], Bulk Synchronous Parallel ML (BSML) , Multicore-BSP [123][124].	Google File System GFS [192], Yang et al. [200], Map-iterative-reduce [202], Ekanayake et al. [197], Bu et al. [203], Spark [204], Olston et al. [206], SRM [275], iRODS[276], MapReduce-MPI[277], Pilot-MapReduce[278], Lustre [279], GPFS [192], PVS [280]	• Scalability • Programming Abstraction • Exploiting dynamic parallelism • Data locality through abstraction layer • Data centric abstraction. • Heterogeneity
Scheduling and Load Balancing	Ousterhout et al. [99], Falt et al. [103], Muddukrishna et al. [104], Ding et al. [105], Lifflander et al. [106], Xue et al. [107], Isard et al. [108], Maglalang et al. [109], Yoo et al[110], Paidel et al. [111], Guo Yi [113], Hindman et al. [114], Isard et al. [115]	Guo et al. [170], Chen et al. [220], Chen et al. [221], Wang et al. [102], Park et al. [222], Zaharia et al. [223], Zhang et al. [224], Hsu et al. [225], Xue et al. [226], Sadasivam et al. [227], Zhang et al. [228], Guo et al. [229], Hammoud et al. [230], Ahmad et al. [231], Kumar et al. [232], Zhao et al. [233], Hammoud et al. [234], Ibrahim et al. [235], Mozakka et al. [37], Sethi et al. [236], Yang et al. [237], Bezerra et al. [238]	• Poor scalability • Heterogeneity • Locality aware scheduling algorithms • Portability • Complexity
Parallelism Mapping	Jeannot et al. [281], Rashti et al. [139], Hestness et al. [140], HU chen et al. [141], Zhang et al. [142], Zarrinchain et al. [144], Guillaume et al. [145], Blue Gene systems [146][147][148], multicore networks [139][149][150], hybrid MPI/OpenMP mapping [151], mapping library [152], Grewe et al. [156], Tournavitis et al. [157], Wang et al. [158]	Map-Reduce [191], Hadoop [193], Map-iterative-reduce [202], Spark [204], Engle et al. [244], Olston et al. [206].	• Manual Mapping is time consuming and error prone • Portability • High Complexity (Compiler based techniques) • Expensive compilation Overhead • Lack of Intelligence
In situ Data Analysis	Tiwari et al. [165], Zheng et al. [166], Sewell et al. [167], Sriram et al. [168], Zou et al. [169], Kim et al. [170], Sriram et al. [171], Yu Su et al. [172], Karimabadi et al[173], Yu et al. [174], Zou et al. [175], Woodring et al. [176], Nouanesengsy et al. [177], Landge et al. [178], Zhang et al. [179]	Wang et al. [282], Xu et al. [283], [165], Wang et al. [282], Xu et al. [283], Spark on demand [284].	• Data Size • Energy Efficiency • Resource utilization • Limited storage and bandwidth • Data Movement cost • Efficient Data Analysis • Compression/decompression overhead • Indexing (compute and memory intensive) • I/O issues • In-situ visualization
Locality aware partioning	Zhang et al. [33], NUMA data shuffling [257], data partitioning [254][255][256], NVRAM Memristive devices[259] STT-MRAM[260],	Lin et al. [207], Ibrahim et al. [208], Rhine et al. [209]	• Lack of Intelligence • Complexity • Load balancing • Data Dependencies
Data Placement		Eltabakh et al. [161], Yu et al. [162], Tan et al. [163], Wang et al. [164], Xie et al. [214], Arasanal et al. [215], Wei Lee [216], Ubarhande et al. [217], Sujitha et al. [218]	• Locality aware storage • Communication •
In-Memory computation	Sun et al. [239], Shen Li et al. [243], Engle et al. [244], Sentos-Neto et al. [245], Reynold et al. [246], In Memory Data Processing Systems		• Emerging new non-volatile memory technologies • Network/storage aggregation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

	Spark [240], Mammoth [261], Piccolo [262], S4 [263], Map-reduce online [264].	• Efficient utilization of cache • Optimized utilization of storage management
Cache Centric Optimization	Compression [247], colouring [248], decomposition storage model [251], re-organizing data layouts [252][253], Gupta et al. [73], Gonzalez et al. [74], on-chip caching [75][76], data access frequency [77][78], Kennedy et al. [93], Sparsh Mittal [94].	• Efficient exploitation of cache • Cache aware partitioning • Smart, dynamic and predictive optimizations.

5. Data Locality in Converged HPC and Big Data Environments

Now that we have reviewed the literature on data locality in HPC and big data environments, in this section we review the literature on converged systems. We first capture in **Error! Reference source not found.** a summary of the efforts in big data and HPC that were reviewed in Sections 3 and 4. Specifically, the table gives a brief overview of research efforts related to data locality at different levels of the software ecosystem and also highlights some of the convergence challenges.

In a typical HPC environment both compute and storage servers are separated and the cost of moving these large data sets is very high. The High-end computation machine and storage cluster running parallel file system are connected with high-speed network. Data intensive applications in this setup demands high data movement across network, which is a major bottleneck. In contrast to big data paradigm data management in HPC environment lacks higher-level abstraction [44]. Solutions have emerged over the years to deal with massive amount of data in data intensive applications e.g. SRM [275], iRODS [276], MapReduce-MPI [277], Pilot-MapReduce [278] etc.

HPC applications uses parallel programing paradigm such as MPI to exploit parallelism, rely on low latency network for message passing and uses parallel file system for example Lustre [279], GPFS [192], PVS [280] etc. Data intensive computing makes use of distributed file systems, which includes Google file system GFS [192], HDFS Hadoop distributed file system [194] etc. The HPC applications uses data intensive distributed file system through an interface for example libHDFS [285]. Although these file systems are tailored for different targeted applications and computing environment but they have somehow identical abstract level designs [286]. Data consistency is not a priority for data intensive file system and is usually compromised for better performance by introducing client-side cache to improve bandwidth. Parallel file systems support concurrency while cache coherency is maintained in data intensive file system through data locking techniques. The client and server process collocated for enhanced I/O performance while data locality is not prime design choice for parallel file systems [44].

A discussed before data locality is considered as a major concern for optimized data movement and co-location of computing and data to reduce communication between process/threads/compute nodes to achieve energy efficient exascale computing. There is huge body of work related to data locality and have been presented in the previous section for both big data and HPC domain. Following sections discusses the research efforts, which can be considered as baby steps towards data locality aware convergence of HPC and big data.

5.1. MPI with Map-Reduce Frameworks

MPI is a de facto standard and is widely used in high performance computing environment for effective and efficient communication. Researchers have experimented with idea of using MPI for data intensive computing and somehow succeeded. Hoefler et al. [287] proposed a scalable implementation of map-reduce functionality using MPI and also proposed numerous possible extensions in MPI to support map-reduce. Hadoop map-reduce provides fault tolerance through redundant storage and reallocation of work where as MPI based implementations are deprived of that. The MPI –map-reduce integration can provide efficient implementation with optimized data movement by controlling

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

where data resides for each map and reduce phase which can be achieved by user defined hash functions [277].

Parallelization of many task applications have been tried with different workflow systems e.g. MPI, ad-hoc Hadoop [193], CloudBlast [288] , Spark [240] and HTCondor [289]. Zhang et al. [290] used Apache Spark to parallelize many task applications by using Kira (an astronomy image processing toolkit) and compared its performance with equivalent parallel implementation using HPC toolset and improved performance is reported. Lu et al. [291] identified the challenges and potential benefits of reducing communication overhead by using MPI with map-reduce and also highlighted the possible MPI extensions for optimized integrated MPI-map-reduce programming paradigms. DataMPI exploits the overlapping of map, shuffle and merge phases of map-reduce framework and increases data locality during the reduce phase. It provides the best performance and an average energy efficiency [292]. Mohamed et al. [293] proposed overlapping of map and reduce phases by running them concurrently and MPI is used as a message passing communication medium between the two, to exchange partial intermediate data.

5.2. Map-Reduce Frameworks with High Performance Interconnects

Data intensive applications have been extensively used in HPC infrastructure with multicore systems using Map-reduce programming model [294]. Hadoop relies on legacy TCP/IP protocols for the transferring of intermediate data, which makes Hadoop incapable of utilizing the benefits of RDMA. So it is difficult to use high performance interconnects in an optimal way so different HPC oriented map-reduce solutions came that addresses the problem of leveraging high performance interconnects [295] i.e. RDMA –Hadoop, DataMPI [296] etc. Hadoop has its own limitations of disk and network bandwidth and network bandwidth is increased with use of infiniband. TCP/IP protocol is used as means of communication protocol in Hadoop through Java sockets [292]. Different solutions emerged to address this problem for efficient use of map-reduce with high performance interconnects.

Yandang et al. [297] presented a comparative analysis of infiniband and 10GigaBit and performance of both is evaluated on Hadoop. Performance is considerably improved when the intermediate data size is small while with large intermediate data size performance degradation is reported. Disk bottleneck and scalability also improved with use of Hadoop with high performance interconnect. Yu et al. [298] proposed an acceleration framework to optimize Hadoop for fast data movement and also proposed network levitated merge algorithm. Reduce task gets the intermediate data from Map output and stores it locally on to memory, which leads to multiple disk access and I/O operations. The proposed algorithm overcomes this by fetching only a header of the segment instead of fetching whole segment.

Dhabaleshwar. K Panda [299] emphasizes on the effectiveness of using infiniband in terms of cost for large scale clusters then its counterpart, the standard Ethernet. Most of the HPC based map-reduce solutions (RDMA, DataMPI [296], HMOR) are affected by the degree of change in default Hadoop framework to exploit benefits of high speed interconnects but Mellanox UDA [300][301] and IP over InfiniBand (IPoIB) [302] requires minimum to no changes to Hadoop configuration. Hadoop is linear scalable and with increasing size of clusters, organizations started using infiniband and solid-state drives (SSDs). InfiniBand along with RDMA achieved almost 4 times bandwidth compared to 10GigaBit Ethernet port can deliver [299].

According to project Aloja [303], there are numerous Hadoop's performance tuneable parameters like Hardware, RAM capacity, storage type, HDFS block size, number of mappers and reducers, network speed etc. According to their findings, adding infiniBand does not improve the performance but using it with solid state drives (SSDs) delivered 3.5X performance enhancement compared to Gigabit Ethernet. Islam et al. [304] identifies different challenges of pipe-lined replication scheme and proposes an alternative parallel replication scheme and compared its performance with existing pipelined replication in

 doi:10.20944/preprints202211.0161.v1

HDFS over Ethernet, IPoIB, 10 GigE and RDMA and showed performance enhancement with parallel model for large data sizes and high performance interconnects.

Lu et al. [295] highlighted the potential benefits of integrating Spark and RDMA framework and proposed a RDMA based solution to accelerate a data shuffling in Spark by using high performance interconnects. On the similar line Wasi-ur-Rehman et al. [305] proposed Hadoop map-reduce over infiniBand using RDMA, Islam et al. [306] presented HDFS with RDMA over Infiniband and Lu et al. [307] proposed Hadoop RDMA based Hadoop RPC over infiniBand.

5.3. Map-Reduce like framework for in-situ analysis

Scientific applications often run on high-performance computing cluster, followed by offline data analysis tasks on smaller clusters. Expensive CPU hours on High End Computing (HEC) machine is one of the main reasons for this offline cluster analysis. So, compute intensive simulations runs on HEC machine and data analysis task is performed on smaller cluster after the completion of simulation. This approach has several disadvantages in terms of performance; energy consumption and redundant I/O, which ultimately resulted in, increase data traffic between compute and storage subsystems[165]. Scientific data sets are stored in back end storage servers in HPC environment and these data sets can be analysed by YARN map-reduce program on compute nodes. As both compute and storage servers are separated in HPC environment, the cost of moving these large data sets is very high. Wang et al. [282] proposed a Map-reduce like framework for in situ data analysis which requires minimum modification in simulation code. Compared to traditional Map-reduce their system performs analysis task by fetching data directly from memory in each node and keep the memory utilization low by avoiding key-value pair output. They evaluated the system by using different scientific simulations on both multicore and many core clusters with minimum overhead. Although many HPC systems have exploited in situ data analysis but there is still a need for the efficient analysis of data stored in backend storage system. Xu et al. [283] proposed virtualised Analytics Shipping (VAS) framework with fast network and disk I/O for efficient shipping of Map-reduce programs to Lustre storage servers**.** Spark on demand allows users to use Apache Spark for in situ data analysis of big data on HPC resources [284]. With this setup, there is no longer to move petabytes of data for advance data analytics. **Error! Reference source not found.** summarizes the research efforts related to the convergence of HPC and bigdata along with the challenges and future directions.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

Table 2. Data Locality Aware HPC and Bigdata Convergence Efforts.

Convergence	Convergence Efforts	Challenges/ Future Directions
MPI with MAP-Reduce	Hoefler et al. [287], MPI, ad-hoc Hadoop [193], CloudBlast [288], HTCondor [289], Zhang et al. [290], Lu et al. [291], DataMPI [292], Mohamed et al. [293], Pilot-Jobs [308], Pregel [118], Apache Hama [119] and Giraph [120], SRM [275], iRODS [276], MapReduce-MPI [277], Pilot-MapReduce [278].	• Programming Abstraction • Minimizing Complexity (Degree of change of default MapReduce). • Improving Parallel Replication Scheme • Adaptability • Innovation in Data placement and data access strategies • Improving data layout strategies
Map-Reduce with High performance Interconnects	DataMPI [296], [240], Yandang et al. [297], Yu et al. [298], Dhabaleshwar. K Panda [299], Mellanox UDA [300][301], IP over InfiniBand (IPoIB) [302], Aloja [303], Islam et al. [304], Lu et al. [295], Wasi-ur-Rehman et al. [305], Islam et al. [306], Lu et al. [307]	• Scalability • Complexity • High Bandwidth and Low latency interconnects • Efficient Data transfer • Energy Efficiency • Compatibility • S/w H/W co-design
In situ Analysis	Wang et al. [282], . Xu et al. [283], Spark on demand [284].	• Data Volume • Data Management • Energy Efficiency • Complexity • Cognitive computing and storage

6. Challenges, Opportunities, and Future Directions

HPC and Big data are different paradigms (Compute centric vs. data centric) but also have different software echo system. The convergence of both these paradigms demands collaborative efforts at different levels of their eco-systems. Hadoop relatively new but matured over the years and started supporting different heterogeneous workloads [44] especially with introduction of YARN [309] and Mesos [114]. Pilot-Jobs [308] and other tools emerged for data intensive jobs in HPC environment but lack the scalability of Hadoop [44]. The following table summarizes the difference between big data (Hadoop) and HPC eco system.

Table 3. HPC vs. Hadoop Eco Systems.

	Big Data	HPC
Programming Model	Java Applications, SparQL	Fortran, C, C++
High level Programming	Pig, Hive, Drill	Domain specific Language
Parallel run time	Map-reduce	MPI, Open MP, OpenCL
Data Management	HBase, MySQL	iRODS
Scheduling (Resource Management)	YARN	SLRUM (Simple LINUX utility for resource management)
File system	HDFS, SPARK (Local storage)	LUSTRE (Remote storage)
Storage	Local shared nothing architecture	Remote shared parallel storage
Hardware for Storage	HDDS	SSD
Interconnect	Switch Ethernet	Switch Fiber
Infrastructure	Cloud	Supercomputer

As we are heading towards exascale systems, achieving billion-fold parallelism within energy constraints is extremely challenging task. The outburst of data being produced at brisk speed brings many challenges that may include but are not limited to minimized data movement, data locality, data storage, effective and efficient searching algorithms and data analysis.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

Reed et al. [18] identified several exascale challenges including 1) high bandwidth and low latency interconnect technologies that also requires locality aware algorithms for efficient data transfer. 2) Advancement in memory technology that directly influences data movement and energy constraints. 3) Data management software to handle massive amount of data and efficient in situ data analysis requires some revolutionary changes in applications and scientific workflows .4) programming models for expressing parallelism and data locality, alleviating programmers' burden of expressing billion-fold parallelism and fault handling.

Data locality thus is coined as one of the major issue research communities is facing for exascale systems and currently managed in Peta-scale systems at application level, file system or middle level. Parallel file systems e.g. Lustre [279], GPFS [192], PVS [280] are bound to be compatible with POSIX (Portable Operating System Interface) for maintaining compatibility between operating systems but compromise data locality. On the other hand, Google File System (GFS) use map-reduce processing framework to avoid POSIX constrained. These frameworks brought in best efforts to bring computation to data and are widely used in data intensive applications. These techniques tailored for data centric computation are not well suited therefore, not widely adopted by the HPC community [310]. The run time performance depends on efficient task scheduling by optimally allocating tasks to the target architecture. Over provisioning of parallel work leads to threads spending a long time in the waiting queues for required resources and under provisioning leads to underutilization of resources. Locality aware performance optimization of an application on multicore architecture is challenging due to shared, hybrid and distributed memory architecture with several hierarchies determined by non-uniform communication latencies.

6.1. Programming paradigms

Most of the current programming paradigms in HPC and big data environment (MPI, OpenMP, OpenCL, Map-Reduce) does not meet the needs of exascale computing and demands thorough reinvestigation [311]. Hybrid approaches also needs some innovation more specifically on locality-based communication to address scalability and billion-fold parallelism. Partitioned Global Address Space PGAS model offers rich set of functionalities and its different implementations present multi-threaded view while MPI depicts fragmented data view. Load balancing, increasing complexity with scalability and lack of hierarchical decomposition are some of the constraints of PGAS which limited its growth to address issues related to converged exascale system [310]. OpenCL can be a potential candidate due to its portability but is criticized for being too low level, leaving complexity to programmer to handle data transfer, synchronization etc. None of the programing paradigms actually fits in exascale era and needs to build new programming model to overcome the limitation of current programing infrastructures. There is also a need for efficient data placement and data access strategies to reduce communication and cost of data movement. The complexities to handle these low-level details compelled engineers to focus on high-level optimization by focusing more on minimizing communication.

6.2. Programming models and Language support

The evolution of software productivity could not match the speed, at which hardware and network technologies have evolved over the years. There is a need for some evolutionary and revolutionary changes at different levels of software eco system to address the issues related to converged HPC and big data environment. Performance optimization in terms Gflops/sec is no more viable in today's world as energy consumption is one of the primary concerns for exascale systems. Energy efficiency is directly affected by data locality which in turn can be achieved by bringing computation to data, minimizing data movement by efficient exploitation of cache hierarchies, reducing communication, locality aware process/thread mapping and in-situ/in-transit data analysis. There is a need to invest considerable efforts for investigating locality aware programming models with

 doi:10.20944/preprints202211.0161.v1

compiler support, runtime environments, high level languages and abstractions strategies to build flexible, dynamic systems to capitalize multi-many core architectures with complex memory hierarchies. There is trade-off between performance and portability, which needs to be clearly understood. Compilers need innovations and runtime system needs intelligence for efficient exploitation of data locality and performance at runtime. At application level algorithms must address data locality, load balancing, scalability and communication [62]. The application and data interoperability issue between different programming models/languages also needs further investigation.

6.3. Programming Abstractions

There is need for the efficient high-level programming, exploiting dynamic parallelism, data locality through abstraction layer and complex structure abstractions [27] to facilitate and assist programmers to address data locality issues and resource abstractions (from physical resources i.e. memory, cores) for effective resource utilizations in converged HPC and big data environment. Data centric abstraction exploitation is needed for the converged HPC and big data environment to alleviate the programmer's burden to deal with heterogeneous systems.

6.4. Innovations in data layout strategies

Data locality issues must be addressed at different layers of Input/output stack that demands strategic, dynamic adoptive and predictive methodologies to collocate the computation and data [310]. There is also a need for Locality aware data distribution based on the runtime behavior of the application, which is unknown at, compile time. One way of addressing this problem is the use of machine learning based techniques to predict the runtime behavior by capitalizing low overhead profiling tools to extract the runtime features (runtime data mobility from backend storage to application) that can be used to train the predictor for optimal code/core selection to minimize communication. Intelligent data placement algorithms need to be investigated for reducing un-necessary communication.

6.5. Locality-Aware Scheduling

As the number computing resources is on a rise with complex memory hierarchies and heterogeneity making locality aware scheduling and resource management a challenging task and cannot be handled efficiently by the current centralized scheduling systems. There is need for adoptive distributed job scheduling management infrastructure, re-engineering of locality aware scheduling algorithms and amalgamation of data locality with allocation mechanism but without compromising the scalability and energy constraints. AI-assisted approaches such as [57], [312], [313] would play an important role n this direction.

6.6. Software Hardware Co-Design

There is need for coordinated efforts in Software hardware co-design to address the challenges of future generation systems with ability to handle applications from multi-dimensional domains. The mapping of application and system software should also be aided by the co-design [62] so that data locality issue can be addressed by minimizing communication.

6.7. Innovations in Memory and storage Technologies

Minimize data movement demands innovation in memory technologies. The classical DRAM and SRAM may not be suitable for future systems that result in emerging new non-volatile memory technologies but still at their infancy stages. There is need for the integrated software stack to addresses the issues of I/O requirements of data centric applications via network/storage aggregation, efficient utilization of client-side cache along with server-side optimized utilization of storage management.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

6.8. High-speed Interconnects

The scalability of communication bandwidth of high-speed interconnects must match with increasing processing capability of a node with multi-many cores. The compact integration of interconnects must minimize the remote data access latency by providing high bandwidth and low latency efficient interconnection with in energy constraints.

7. Proposed Future (HPC and Big data Converged System Architecture)

As the complexity of the computer systems is on the rise with the number of cores per processors, different levels of cache, processors per node and high-speed interconnects, there is an ever-growing need for the new optimization techniques to minimize communication and efficient way of exploiting parallelism for heterogeneous resources. Also, computation is getting cheaper; there is a need for a paradigm shift from compute centric to data centric. The current programming environments do not fully facilitate the abstraction to optimize data locality resulting in programmers to use other means for locality aware optimization. There is a potential scope to exploit locality by providing eco-friendly, environmentally responsible computations and innovations in algorithmic research i.e. communication avoiding algorithms and a need for automated parallelism mapping on target system software and hardware architecture [62].

Implementing codes with system characteristics in mind (minimising communication, avoid cache misses by using blocks that fits in the cache) and mapping processes to specific cores are the two main approaches studied to improve the performance of parallel application in multicore architecture. The optimal task mapping requires heuristics to find best mapping strategy but these heuristics requires run time behaviour of these tasks to be known that in turns requires static and dynamic analysis to facilitate automatic mapping. One of the major challenges to map the parallel executable tasks is to obtain the needed run time behaviour of these tasks.

One way of looking at data locality issues in converged HPC and big data environments is by effectively and efficiently utilizing resources that should ultimately minimize communication. Once the code is parallelized, there is need for the efficient code mapping to underlying architecture for efficient exploitation of hardware resources. Optimal mapping decision is non-trivial and depends not only on parallel algorithm but also on relative cost of communication and computation. Expert programmers can implement effective mapping but manual process is expensive and error prone. Although manual mapping can be effective on a particular architecture where programmer is responsible for all the issues i.e. load-balancing, synchronization, communication etc. but this solution is not portable and needs considerable efforts for the code to run on different platform [158].

Compiler based mapping techniques can be an alternative for parallelism mapping but manual tuning of compiler-based approaches are complex to handle with increasing level of hardware complexity and application diversity. Researchers engineered numerous compiler-based heuristics (mostly platform specific) over the years, based on analytical models to optimize compilation decisions. Feedback directed or iterative compiling produces multiple version of program and empirically measures the performance by actually running codes on target hardware. These measured performance act as a feedback for the selection of best among different options. This does require exhaustive search space for optimal solution with expensive compilation overhead. To overcome these issues predictive modelling came into picture but much of the work in literature is based on sequential programs [158] [314]. Predictive techniques predict optimization (without executing them on target hardware or simulator) based on previous knowledge generated by offline training with different compilation options which are then used to predict the best among available options. In contrast to static compilation techniques, dynamic adaption makes runtime decisions based on dynamic environment information i.e. number of processing elements, program's execution environment for dynamic scheduling of parallel program.

 doi:10.20944/preprints202211.0161.v1

Exascale systems are expected to have millions of components with deep hierarchical structure both horizontal (network interconnection) and vertical (memory architecture i.e. cache hierarchies, NUMA). To achieve massive scalability there is a need for efficient exploitation of resources but within energy constraints that can be achieved by addressing data locality i.e. how data is placed, accessed and moved across complex system hierarchies (horizontal and vertical). We believe that future-computing systems will require research in three main dimensions: characterization of workloads, characterization of the system resources, and thirdly smart ways to map the workloads to the underlying system resources under multiple constraints with configurable preferences. We have highlighted some of the work related to process mapping by categorizing them as algorithmic approaches and machine learning based techniques. There is a need to dedicate efforts for automated approaches to provide a portable mapping solution, which models the interaction of parallel application, and underlying architecture effectively and efficiently. Machine/deep learning thus can be applied to automate the process of mapping workload to processing cores for an optimal load balancing, scalability and address the locality issues either by avoiding or minimizing communication. Mapping application tasks on multi/many-core system involves assignment and ordering of the tasks and their communications onto the platform resources. The communicating tasks are mapped on the same core or close to each other in order to optimize the communication delay and energy efficiency. There is a need to adapt environmentally friendly green computing practices including green production, recycling, development and design.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

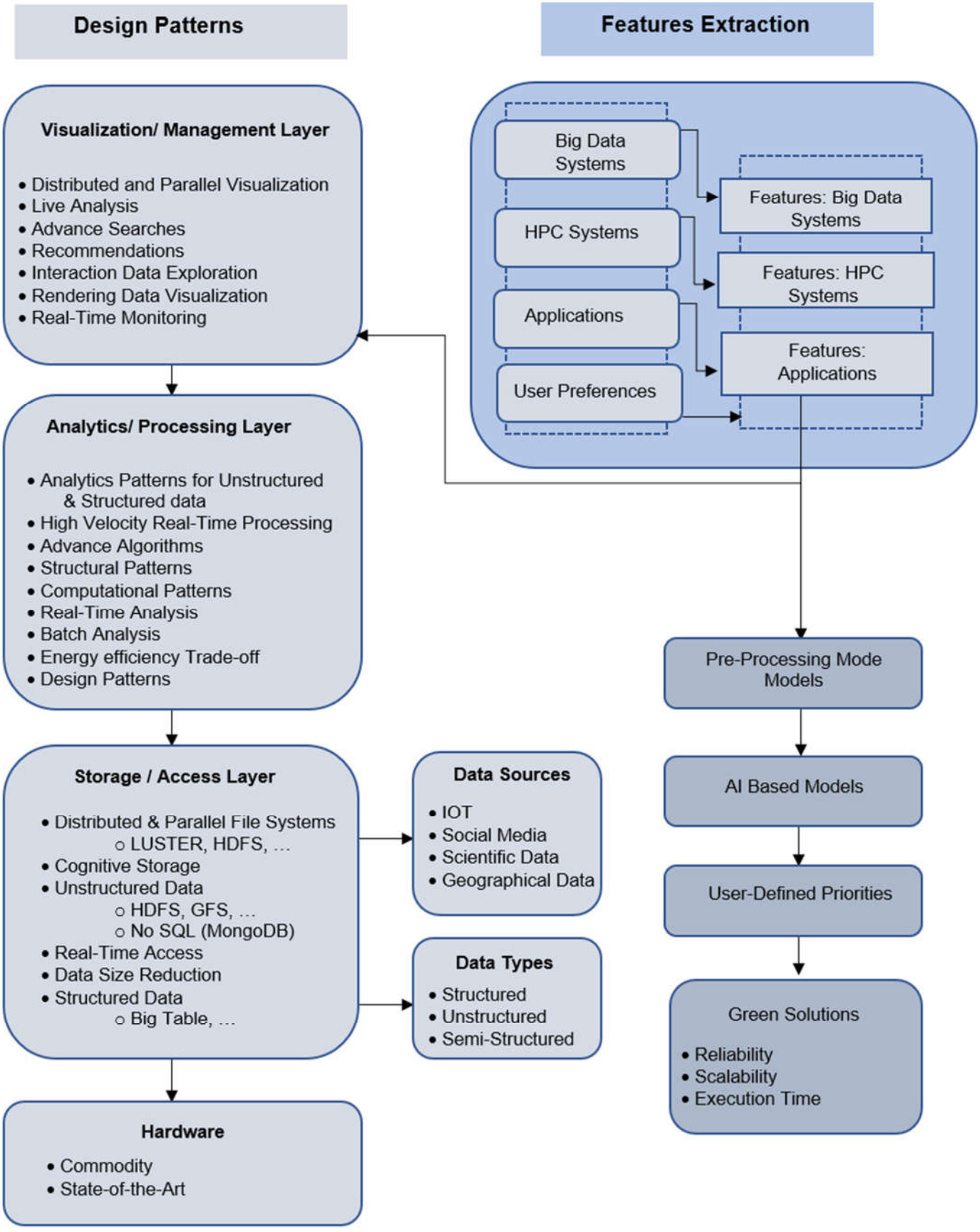

Figure 1. Design patterns and AI-based Architecture for Converged HPC and Big Data Environments.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 November 2022 doi:10.20944/preprints202211.0161.v1

Figure 1 gives an abstract view of machine learning and design patterns-based solution for the converged HPC and big Data environment. Design patterns can be seen as repeatable general solutions to common occurring problems in software design to allow flexible and robust development. The standardization and organization of design patterns in both HPC and big data environments can be seen as potential candidates for this inevitable convergence. These design patterns serve here as a catalyst to produce a dataset which is used for the training of AI based models to address convergence challenges i.e. data locality, energy efficiency, fault tolerance etc. These challenges and growing demands of HPDA to speed up data analysis requires revolutionary and evolutionary changes at different level of HPC and Big Data eco-systems. Generic guide lines provided by these design patterns would help software developers to architect robust and energy efficient solutions. High bandwidth requirements and increasing network complexity further increase the energy consumption and heat emission. So there is need to shift the focus to computing with renewable energy and green solutions for the converged HPC and big data environment.

The feature extractor extracts the required static and dynamic features, which can then be used for preparing training data for AI based models along with optimal solution based on user's preferences. Software processing entities (processes, threads) and their dependencies can be expressed as application virtual topology e.g. message exchanged between process in message passing models (MPI) and access to common memory location (OpenMP). These software-processing entities need to communicate with each other in regular or irregular manner, which demands efficient utilization of available resources to facilitate data access and communication. The characteristics of underlying hardware architecture needs to be gathered in a portable way to target broad range of architectures. The cluster environment demands network topology and multi-many-core environment demands intra-node structure information to be accumulated in a comprehensive way in order to map the application virtual topology on the targeted physical topology. The memory access behaviour for each task needs to be gathered with minimum overhead as many existing memory tracing techniques are based on simulating application with big overhead. Hardware counters are also used for this purpose but have low accuracy due to sampling of applications memory accesses. An alternative way is Dynamic Binary Instrumentation (Valgrind, MemTrace) but also have lack of direct support for parallel applications, overhead and accuracy issues. There is need for more sophisticated tools that provide us with all the necessary information with minimum overhead and maximum accuracy [315].

In distributed environment there is a need for the innovation in the message passing models to effectively utilize the internal communication pipeline based on the underlying network topology. Current supercomputing system use job schedulers for resource allocation (SLRUM, PBS) but does not consider the application communication requirements and lack the intelligence to map tasks to underlying topology. The complexity will increase further with future systems with millions of cores arranged in multiple level of hierarchies, multiple sockets with number of cores, nodes with multiples sockets, blades with multiple nodes, multiple blades arranged in racks, whole system arranged in multiple racks and interconnection of these components with complex network topologies. The issues related to resources contention, routing schemes of underlying network topology, scalability, job scheduling and process mapping, needs further investigation to achieve scalable performance. There is need for the topology aware communication libraries and schedulers to use runtime network information to make intelligent scheduling decisions [316] and also considers simultaneous mapping of concurrent applications to heterogeneous resources.

The use of machine learning helps developers to automatically engineer dynamic optimization strategies and runtime adaption methodologies to cater changing program behaviours. Run time adaptions can be facilitated by dynamic configuration of hardware using machine-learning techniques. Capturing the static program features along with dynamic features can be used to predict the applications behaviour, which can then be used

 doi:10.20944/preprints202211.0161.v1

to configure the hardware resources. There is a need to find the synergy between high-level optimizations for parallelism mapping and low- level compiler transformations by carefully considering the trade-offs for an optimized mapping [317].

Design patterns thus can be used at different layers of software eco system to address the broad range of challenges including scalability, elasticity, adaptability, robustness, locality and storage with solutions that are already tested and implemented in similar or closely related environment e.g. parallel design patterns (Our Pattern Language OPL) [318], Big-data patterns [319]. Both OPL and Big-data patterns are organised in a logical architecture of different layers. Figure 1 shows the layered architecture of design patterns which act as a catalyst to produce high quality data for AI based models to address different problems based on user's preferences including fault tolerance, parallelization mapping which ultimately improves data locality, energy efficiency thus providing green solutions for the converged HPC and Big data environment. Applying multiple design patterns is itself a tedious task due to diverse nature of these design patterns and there are few efforts to address these challenges using machine-learning techniques [320][321]. These design patterns assist (both HPC and big data) software developers to architect and address data locality issues efficiently. Design solutions of data visualization and interactive management are hard to assess and reapply. At the abstract level, different design patterns at visualization and management layer are defined to address distributed, parallel, interactive and live data visualization and analysis. The processing layer includes design patterns, high velocity real-time processing, large scale batch/graph analysis, strategy patterns, data conversion, structural and computational patterns etc. the trade-off between fault tolerance, performance and energy efficiency needs careful monitoring by incorporating best practices from both HPC and Big Data environments. Parallel and distributed design patterns for smart storage and efficient retrieval are organized at storage and access layer as shown in Figure 1. These design patterns provide cognitive storage (automated purging), data size reduction, high volume hierarchical, linked, tabular, and binary storage with indirect and integrated data access. Due to high volume and applying complexity of these design patterns, there is need to focus on exploration of innovative approaches to automate the process of applying these design patterns to address convergence challenges.

The use of AI techniques, standardization and efficient exploitation of design patterns can lead to effective and systematic solution to address the locality and other challenges of converged HPC and big-data environment.

8. Conclusion

This paper presented a review of the cutting-edge on data locality in HPC, big data, and converged systems. The efficient resource utilization and performance of parallel applications demands intelligent parallelism mapping to the target architecture but within energy constraints. Numerous research efforts dedicated to data locality issues at different levels of HPC and big data software eco systems are presented. The paper also provides research efforts for the inevitable convergence of HPC and big data by primarily focusing on locality and highlighted some of the challenges that need further investigation. To address these challenges, we also present future trends and proposed a solution to be considered as a future direction. The outburst of data more specifically in the edge environment from multidisciplinary domains (IoT, smart cities, remote sensors i.e. satellite imagery) demands investing research efforts towards "fog or edge" computing infrastructure to provide storage, processing and communication facilities for the integrated HPC and big data environment. So, there is a need to promote the development of software libraries for intermediate processing. The connected and ubiquitous synergy between HPC and big data demands the exploration of some revolutionary and evolutionary innovations and coordinated efforts at different levels of integrated software eco system.

The use of Design patterns, cognitive computing (Machine learning, natural language processing) and intelligent process mapping can be seen as potential candidates to address the data locality and other challenges for the integrated HPC and big data environment.

Author Contributions: The following statements should be used "conceptualization, S.U. and R.M.; methodology, S.U. and R.M.; validation, S.U., R.M., I.K., and A.A.; formal analysis, S.U., R.M., I.K., and A.A.; investigation, S.U., R.M., I.K., and A.A.; resources, R.M., I.K., and A.A.; data curation, S.U.; writing—original draft preparation, S.U. and R.M.; writing—review and editing, R.M., I.K., and A.A.; visualization, S.U.; supervision, R.M., and I.K.; project administration, R.M., I.K., and A.A.; funding acquisition, R.M., I.K., and A.A.". All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge with thanks the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under Grant No. RG-10-611-38.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not Applicable.

Acknowledgments: The work carried out in this paper is supported by the HPC Center at the King Abdulaziz University.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mob. Networks Appl.*, vol. 19, no. 2, pp. 171–209, 2014, doi: 10.1007/s11036-013-0489-0.

[2] Rob Farber, "The Convergence of Big Data and Extreme-Scale HPC," *HPC Wire*, 2018. https://www.hpcwire.com/2018/08/31/the-convergence-of-big-data-and-extreme-scale-hpc/ (accessed Nov. 01, 2022).

[3] F. Alam, A. Almaghthawi, I. Katib, A. Albeshri, and R. Mehmood, "iResponse: An AI and IoT-Enabled Framework for Autonomous COVID-19 Pandemic Management," *Sustainability*, vol. 13, no. 7, p. 3797, Mar. 2021, doi: 10.3390/su13073797.

[4] E. Alomari, I. Katib, A. Albeshri, T. Yigitcanlar, and R. Mehmood, "Iktishaf+: A Big Data Tool with Automatic Labeling for Road Traffic Social Sensing and Event Detection Using Distributed Machine Learning," *Sensors*, vol. 21, no. 9, p. 2993, Apr. 2021, doi: 10.3390/s21092993.

[5] G. Alkhayat, S. H. Hasan, and R. Mehmood, "SENERGY: A Novel Deep Learning-Based Auto-Selective Approach and Tool for Solar Energy Forecasting," *Energies 2022, Vol. 15, Page 6659*, vol. 15, no. 18, p. 6659, Sep. 2022, doi: 10.3390/EN15186659.

[6] N. Alahmari *et al.*, "Musawah: A Data-Driven AI Approach and Tool to Co-Create Healthcare Services with a Case Study on Cancer Disease in Saudi Arabia," *Sustain. 2022, Vol. 14, Page 3313*, vol. 14, no. 6, p. 3313, Mar. 2022, doi: 10.3390/SU14063313.

[7] S. Alswedani, R. Mehmood, and I. Katib, "Sustainable Participatory Governance: Data-Driven Discovery of Parameters for Planning Online and In-Class Education in Saudi Arabia During COVID-19," *Front. Sustain. Cities*, vol. 0, p. 97, Jul. 2022, doi: 10.3389/FRSC.2022.871171.

[8] A. A. Alaql, F. AlQurashi, and R. Mehmood, "Data-Driven Deep Journalism to Discover Age Dynamics in Multi-Generational Labour Markets from LinkedIn Media," Oct. 2022, doi: 10.20944/PREPRINTS202210.0472.V1.

[9] E. Alqahtani, N. Janbi, S. Sharaf, and R. Mehmood, "Smart Homes and Families to Enable Sustainable Societies: A Data-Driven Approach for Multi-Perspective Parameter Discovery Using BERT Modelling," *Sustain. 2022, Vol. 14, Page 13534*, vol. 14, no. 20, p. 13534, Oct. 2022, doi: 10.3390/SU142013534.

[10] N. Janbi *et al.*, "Imtidad: A Reference Architecture and a Case Study on Developing Distributed AI Services for Skin Disease Diagnosis over Cloud, Fog and Edge," *Sensors 2022, Vol. 22, Page 1854*, vol. 22, no. 5, p. 1854, Feb. 2022, doi: 10.3390/S22051854.

[11] Y. Arfat, S. Usman, R. Mehmood, and I. Katib, "Big data tools, technologies, and applications: A survey," in *Smart Infrastructure and Applications Foundations for Smarter Cities and Societies*, Springer Cham, 2020, pp. 453–490.

[12] R. Mehmood, A. Sheikh, C. Catlett, and I. Chlamtac, "Editorial: Smart Societies, Infrastructure, Systems, Technologies, and Applications," *Mob. Networks Appl. 2022*, vol. 1, pp. 1–5, May 2022, doi: 10.1007/S11036-022-01990-Y.

[13] T. Yigitcanlar, L. Butler, E. Windle, K. C. Desouza, R. Mehmood, and J. M. Corchado, "Can Building 'Artificially Intelligent Cities' Safeguard Humanity from Natural Disasters, Pandemics, and Other Catastrophes? An Urban Scholar's Perspective," *Sensors*, vol. 20, no. 10, p. 2988, May 2020, doi: 10.3390/s20102988.

[14] T. Yigitcanlar, J. M. Corchado, R. Mehmood, R. Y. M. Li, K. Mossberger, and K. Desouza, "Responsible Urban Innovation with Local Government Artificial Intelligence (AI): A Conceptual Framework and Research Agenda," *J. Open Innov. Technol. Mark. Complex.*, vol. 7, no. 1, p. 71, Feb. 2021, doi: 10.3390/joitmc7010071.

[15] T. Yigitcanlar, R. Mehmood, and J. M. Corchado, "Green Artificial Intelligence: Towards an Efficient, Sustainable and Equitable Technology for Smart Cities and Futures," *Sustain. 2021, Vol. 13, Page 8952*, vol. 13, no. 16, p. 8952, Aug. 2021, doi: 10.3390/SU13168952.
[16] R. Alsaigh, R. Mehmood, and I. Katib, "AI Explainability and Governance in Smart Energy Systems: A Review," Oct. 2022, doi: 10.48550/arxiv.2211.00069.
[17] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Commun. ACM*, vol. 63, no. 12, pp. 54–63, Nov. 2020, doi: 10.1145/3381831.
[18] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Commun. ACM*, vol. 58, no. 7, pp. 56–68, Jun. 2015, doi: 10.1145/2699414.
[19] D. Elia, S. Fiore, and G. Aloisio, "Towards HPC and Big Data Analytics Convergence: Design and Experimental Evaluation of a HPDA Framework for eScience at Scale," *IEEE Access*, vol. 9, pp. 73307–73326, 2021.
[20] P. Brox, J. Garcia-Blas, D. E. Singh, and J. Carretero, "DICE: Generic Data Abstraction for Enhancing the Convergence of HPC and Big Data," in *Latin American High Performance Computing Conference*, 2022, pp. 106–119.
[21] S. Hachinger *et al.*, "HPC-Cloud-Big Data Convergent Architectures and Research Data Management: The LEXIS Approach," 2021.
[22] S. Karagiorgou *et al.*, "CYBELE: On the Convergence of HPC, Big Data Services, and AI Technologies," in *HPC, Big Data, and AI Convergence Towards Exascale*, CRC Press, 2022, pp. 240–254.
[23] A. Tzenetopoulos *et al.*, "EVOLVE: towards converging big-data, high-performance and cloud-computing worlds," in *2022 Design, Automation \& Test in Europe Conference \& Exhibition (DATE)*, 2022, pp. 975–980.
[24] J. Ejarque *et al.*, "Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence," *Futur. Gener. Comput. Syst.*, vol. 134, pp. 414–429, 2022.
[25] S. R. Sukumar *et al.*, "The Convergence of HPC, AI and Big Data in Rapid-Response to the COVID-19 Pandemic," in *Smoky Mountains Computational Sciences and Engineering Conference*, 2021, pp. 157–172.
[26] A. Scionti, P. Viviani, G. Vitali, C. Vercellino, and O. Terzo, "Enabling the HPC and Artificial Intelligence Cross-Stack Convergence at the Exascale Level," in *HPC, Big Data, and AI Convergence Towards Exascale*, CRC Press, 2022, pp. 37–58.
[27] D. Unat *et al.*, "Trends in Data Locality Abstractions for HPC Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 3007–3020, Oct. 2017, doi: 10.1109/TPDS.2017.2703149.
[28] T. Mohammed, A. Albeshri, I. Katib, and R. Mehmood, "UbiPriSEQ—Deep reinforcement learning to manage privacy, security, energy, and QoS in 5G IoT hetnets," *Appl. Sci.*, vol. 10, no. 20, 2020, doi: 10.3390/app10207120.
[29] N. Janbi, I. Katib, A. Albeshri, and R. Mehmood, "Distributed Artificial Intelligence-as-a-Service (DAIaaS) for Smarter IoE and 6G Environments," *Sensors*, vol. 20, no. 20, p. 5796, Oct. 2020, doi: 10.3390/s20205796.
[30] C. Caragea *et al.*, "Memory Locality," in *Encyclopedia of Database Systems*, Boston, MA: Springer US, 2009, pp. 1713–1714.
[31] M. Snir and J. Yu, "On the Theory of Spatial and Temporal Locality," 2005.
[32] S. Caíno-Lores and J. Carretero, "A Survey on Data-Centric and Data-Aware Techniques for Large Scale Infrastructures," *Int. J. Comput. Inf. Egineering*, vol. 10, no. 3, 2016.
[33] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang, "In-Memory Big Data Management and Processing: A Survey," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1920–1948, Jul. 2015, doi: 10.1109/TKDE.2015.2427795.
[34] S. Dolev, P. Florissi, E. Gudes, S. Sharma, and I. Singer, "A Survey on Geographically Distributed Big-Data Processing using MapReduce," *IEEE Trans. Big Data*, pp. 1–1, 2017, doi: 10.1109/TBDATA.2017.2723473.
[35] M. Senthilkumar and P. Ilango, "A Survey on Job Scheduling in Big Data," *Cybern. Inf. Technol.*, vol. 16, no. 3, pp. 35–51, Jan. 2016, doi: 10.1515/cait-2016-0033.
[36] M. Idris *et al.*, "Context-aware scheduling in MapReduce: a compact review," *Concurr. Comput. Pr. Exper*, 2015, doi: 10.1002/cpe.3578.
[37] M. Mozakka, F. Safi Esfahani, and M. H. Nadimi, "SURVEY ON ADAPTIVE JOB SCHEDULERS IN MAPREDUCE," *J. Theor. Appl. Inf. Technol.*, vol. 31, no. 663, 2014.
[38] Nagina and S. Dhingra, "Scheduling Algorithms in Big Data: A Survey," *Int. J. Eng. Comput. Sci.*, vol. 5, no. 8, pp. 11737–17743, 2016.
[39] N. Kasiviswanath, P. Chenna Reddy, and P. JNTU Pulivendula Pulivendula India, "A Survey on Big Data Management and Job Scheduling," *Int. J. Comput. Appl.*, vol. 130, no. 13, pp. 975–8887, 2015.
[40] H. Akilandeswari. P Srimathi, "Survey on Task Scheduling in Cloud Environment," *IJCTA*, vol. 9, no. 37, pp. 693–698, 2016.
[41] T. Hoefler, E. Jeannot, G. Mercier, E. Jeannot, and J. Žilinskas, "An Overview of Topology Mapping Algorithms and Techniques in High-Performance Computing," in *High-Performance Computing on Complex Environments*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2014, pp. 73–94.
[42] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems," in *Proceedings of the 50th Annual Design Automation Conference on - DAC '13*, 2013, p. 1, doi: 10.1145/2463209.2488734.
[43] H. Asaadi, D. Khaldi, and B. Chapman, "A Comparative Survey of the HPC and Big Data Paradigms: Analysis and Experiments," in *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2016, pp. 423–432, doi: 10.1109/CLUSTER.2016.21.
[44] S. Jha, J. Qiu, A. Luckow, P. Mantha, and G. C. Fox, "A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures," in *2014 IEEE International Congress on Big Data*, Jun. 2014, pp. 645–652, doi: 10.1109/BigData.Congress.2014.137.
[45] M. Asch *et al.*, "Big data and extreme-scale computing.," *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 4, pp. 435–479, 2018, doi: 10.1177/1094342018778123.

[46] F. Yin and F. Shi, "A Comparative Survey of Big Data Computing and HPC: From a Parallel Programming Model to a Cluster Architecture," *Int. J. Parallel Program.*, vol. 50, no. 1, pp. 27–64, 2022.
[47] M. Golasowski *et al.*, "Toward the Convergence of High-Performance Computing, Cloud, and Big Data Domains," in *HPC, Big Data, and AI Convergence Towards Exascale*, CRC Press, 2022, pp. 1–16.
[48] S. Usman, R. Mehmood, and I. Katib, "Big data and hpc convergence for smart infrastructures: A review and proposed architecture," in *Smart Infrastructure and Applications Foundations for Smarter Cities and Societies*, Springer Cham, 2020, pp. 561–586.
[49] S. Usman, R. Mehmood, and I. Katib, "Big Data and HPC Convergence: The Cutting Edge and Outlook," *Smart Soc. Infrastructure, Technol. Appl. SCITA 2017. Lect. Notes Inst. Comput. Sci. Soc. Informatics Telecommun. Eng. Springer*, vol. 224, pp. 11--26, 2018, doi: 10.1007/978-3-319-94180-6_4.
[50] S. Usman, R. Mehmood, and I. Katib, "HPC & Big Data Convergence: The Cutting Edge & Outlook, Poster presented," 2018.
[51] H. Alotaibi, F. Alsolami, E. Abozinadah, and R. Mehmood, "TAWSEEM: A Deep-Learning-Based Tool for Estimating the Number of Unknown Contributors in DNA Profiling," *Electron. 2022, Vol. 11, Page 548*, vol. 11, no. 4, p. 548, Feb. 2022, doi: 10.3390/ELECTRONICS11040548.
[52] E. Alomari, I. Katib, A. Albeshri, and R. Mehmood, "Covid-19: Detecting government pandemic measures and public concerns from twitter arabic data using distributed machine learning," *Int. J. Environ. Res. Public Health*, vol. 18, no. 1, pp. 1–36, Jan. 2021, doi: 10.3390/ijerph18010282.
[53] T. Muhammed, R. Mehmood, A. Albeshri, and F. Alsolami, "HPC-Smart Infrastructures: A Review and Outlook on Performance Analysis Methods and Tools," 2020, pp. 427–451.
[54] M. Aqib, R. Mehmood, A. Alzahrani, I. Katib, A. Albeshri, and S. M. Altowaijri, "Smarter Traffic Prediction Using Big Data, In-Memory Computing, Deep Learning and GPUs," *Sensors*, vol. 19, no. 9, p. 2206, May 2019, doi: 10.3390/s19092206.
[55] T. Muhammed, R. Mehmood, A. Albeshri, and I. Katib, "UbeHealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities," *IEEE Access*, vol. 6, pp. 32258–32285, 2018, doi: 10.1109/ACCESS.2018.2846609.
[56] S. AlAhmadi, T. Muhammed, R. Mehmood, and A. Albeshri, "Performance Characteristics for Sparse Matrix-Vector Multiplication on GPUs," 2020, pp. 409–426.
[57] T. Mohammed, A. Albeshri, I. Katib, and R. Mehmood, "DIESEL: A Novel Deep Learning based Tool for SpMV Computations and Solving Sparse Linear Equation Systems," *J. Supercomput.*, 2020, doi: https://doi.org/10.1007/s11227-020-03489-3.
[58] T. Muhammed, R. Mehmood, A. Albeshri, and I. Katib, "SURAA: A Novel Method and Tool for Loadbalanced and Coalesced SpMV Computations on GPUs," *Appl. Sci.*, vol. 9, no. 5, p. 947, Mar. 2019, doi: 10.3390/app9050947.
[59] S. AlAhmadi, T. Mohammed, A. Albeshri, I. Katib, and R. Mehmood, "Performance Analysis of Sparse Matrix-Vector Multiplication (SpMV) on Graphics Processing Units (GPUs)," *Electronics*, vol. 9, no. 10, p. 1675, Oct. 2020, doi: 10.3390/electronics9101675.
[60] H. Alyahya, R. Mehmood, and I. Katib, "Parallel Iterative Solution of Large Sparse Linear Equation Systems on the Intel MIC Architecture," 2020, pp. 377–407.
[61] R. Mehmood and J. Crowcroft, "Parallel iterative solution method for large sparse linear equation systems," Technical Report Number UCAM-CL-TR-650, Computer Laboratory, University of Cambridge, Cambridge, UK, 2005. Accessed: Feb. 26, 2016. [Online]. Available: https://www.cl.cam.ac.uk/research/srg/netos/papers/MC05.pdf.
[62] "Towards a Breakthrough in Software for Advanced Computing Systems Report from a Workshop organised by the European Commission in preparation for HORIZON 2020 Towards a Breakthrough in Software for Advanced Computing Systems," Brussels, 2012.
[63] Martyn Guest, "The Scientific Case for High Performance Computing in Europe 2012-2020," 2012.
[64] S. Matsuoka *et al.*, "Extreme Big Data (EBD): Next Generation Big Data Infrastructure Technologies Towards Yottabyte/Year," doi: 10.14529/jsfi140206.
[65] "EuropEan TEchnology plaTform for high pErformancE compuTing ETp4hpc Strategic research agenda achieving hpc leadership in Europe," Barcelona, 2013.
[66] T. Hoefler, E. Jeannot, and G. Mercier, "AN OVERVIEW OF TOPOLOGY MAPPING ALGORITHMS AND TECHNIQUES IN HIGH-PERFORMANCE COMPUTING," *hal.archives-ouvertes.fr*, 2013.
[67] Z. Majo, T. R. Gross, Z. Majo, and T. R. Gross, "A library for portable and composable data locality optimizations for NUMA systems," in *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming - PPoPP 2015*, 2015, vol. 50, no. 8, pp. 227–238, doi: 10.1145/2688500.2688509.
[68] C. Lezos, I. Latifis, G. Dimitroulakos, and K. Masselos, "Compiler-Directed Data Locality Optimization in MATLAB," in *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems - SCOPES '16*, 2016, pp. 6–9, doi: 10.1145/2906363.2906378.
[69] J. Ragan-Kelley *et al.*, "Halide," *ACM SIGPLAN Not.*, vol. 48, no. 6, p. 519, Jun. 2013, doi: 10.1145/2499370.2462176.
[70] Brad Chamberlain, "Parallel Processing Languages: Cray's Chapel Programming," 2013. .
[71] P. Charles *et al.*, "X10," in *Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming systems languages and applications - OOPSLA '05*, 2005, vol. 40, no. 10, p. 519, doi: 10.1145/1094811.1094852.
[72] L. Huang, H. Jin, L. Yi, and B. Chapman, "Enabling Locality-Aware Computations in OpenMP," *Sci. Program.*, vol. 18, no. 3–4, pp. 169–181, doi: 10.3233/SPR-2010-0307.
[73] S. Gupta and H. Zhou, "Spatial Locality-Aware Cache Partitioning for Effective Cache Sharing," in *2015 44th International Conference on Parallel Processing*, Sep. 2015, pp. 150–159, doi: 10.1109/ICPP.2015.24.

[74] A. González, C. Aliagas, and M. Valero, "A data cache with multiple caching strategies tuned to different types of locality," in *Proceedings of the 9th international conference on Supercomputing - ICS '95*, 1995, pp. 338–347, doi: 10.1145/224538.224622.
[75] V. Seshadri, O. Mutlu, M. A. Kozuch, and T. C. Mowry, "The evicted-address filter," in *Proceedings of the 21st international conference on Parallel architectures and compilation techniques - PACT '12*, 2012, p. 355, doi: 10.1145/2370816.2370868.
[76] J. A. Rivers and E. S. Davidson, "Reducing conflicts in direct-mapped caches with a temporality-based design," in *Proceedings of the 1996 ICPP Workshop on Challenges for Parallel Processing*, vol. 1, pp. 154–163, doi: 10.1109/ICPP.1996.537156.
[77] T. L. Johnson, W. W. Hwu, T. L. Johnson, and W. W. Hwu, "Run-time adaptive cache hierarchy management via reference analysis," in *Proceedings of the 24th annual international symposium on Computer architecture - ISCA '97*, 1997, vol. 25, no. 2, pp. 315–326, doi: 10.1145/264107.264213.
[78] X. Jiang *et al.*, "CHOP: Adaptive filter-based DRAM caching for CMP server platforms," in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, Jan. 2010, pp. 1–12, doi: 10.1109/HPCA.2010.5416642.
[79] S. S. Muchnick and S. S., *Advanced compiler design and implementation*. Morgan Kaufmann Publishers, 1997.
[80] R. Allen and K. Kennedy, *Optimizing compilers for modern architectures : a dependence-based approach*. Morgan Kaufmann Publishers, 2001.
[81] M. Wolfe, "Loops skewing: The wavefront method revisited," *Int. J. Parallel Program.*, vol. 15, no. 4, pp. 279–293, Aug. 1986, doi: 10.1007/BF01407876.
[82] M. Kowarschik and C. Weiß, "An Overview of Cache Optimization Techniques and Cache-Aware Numerical Algorithms," Springer, Berlin, Heidelberg, 2003, pp. 213–232.
[83] J. Xue and Jingling, *Loop tiling for parallelism*. Kluwer Academic, 2000.
[84] Bin Bao and Chen Ding, "Defensive loop tiling for shared cache," in *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, Feb. 2013, pp. 1–11, doi: 10.1109/CGO.2013.6495008.
[85] M. E. Wolf, M. S. Lam, M. E. Wolf, and M. S. Lam, "A data locality optimizing algorithm," in *Proceedings of the ACM SIGPLAN 1991 conference on Programming language design and implementation - PLDI '91*, 1991, vol. 26, no. 6, pp. 30–44, doi: 10.1145/113445.113449.
[86] F. Irigoin and R. Triolet, "Supernode partitioning," in *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL '88*, 1988, pp. 319–329, doi: 10.1145/73560.73588.
[87] X. Zhou, J.-P. Giacalone, M. J. Garzarán, R. H. Kuhn, Y. Ni, and D. Padua, "Hierarchical overlapped tiling," in *Proceedings of the Tenth International Symposium on Code Generation and Optimization - CHO '12*, 2012, p. 207, doi: 10.1145/2259016.2259044.
[88] L. Liu, L. Chen, C. Wu, and X. Feng, "Global Tiling for Communication Minimal Parallelization on Distributed Memory Systems," in *Euro-Par 2008 – Parallel Processing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 382–391.
[89] K. Hogstedt, L. Carter, and J. Ferrante, "On the parallel execution time of tiled loops," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 3, pp. 307–321, Mar. 2003, doi: 10.1109/TPDS.2003.1189587.
[90] Q. Yi, "Automated programmable control and parameterization of compiler optimizations," in *International Symposium on Code Generation and Optimization (CGO 2011)*, Apr. 2011, pp. 97–106, doi: 10.1109/CGO.2011.5764678.
[91] M. Hall, J. Chame, C. Chen, J. Shin, G. Rudy, and M. M. Khan, "Loop Transformation Recipes for Code Generation and Auto-Tuning," Springer, Berlin, Heidelberg, 2010, pp. 50–64.
[92] S. Tavarageri, L.-N. Pouchet, J. Ramanujam, A. Rountev, and P. Sadayappan, "Dynamic selection of tile sizes," in *2011 18th International Conference on High Performance Computing*, Dec. 2011, pp. 1–10, doi: 10.1109/HiPC.2011.6152742.
[93] K. Kennedy and K. S. McKinley, "Optimizing for parallelism and data locality," in *25th Anniversary International Conference on Supercomputing Anniversary Volume -*, 2014, pp. 151–162, doi: 10.1145/2591635.2667164.
[94] S. Mittal, "A Survey Of Cache Bypassing Techniques," *MDPI J. Low Power Electron. Appl. (Special issue Energy-efficient Scalable Embed. Memories Futur. Technol.*, vol. 6, no. 2, 2016.
[95] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde, "Falkon," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing - SC '07*, 2007, p. 1, doi: 10.1145/1362622.1362680.
[96] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," Springer, Berlin, Heidelberg, 2003, pp. 44–60.
[97] W. Gentzsch, "Sun Grid Engine: towards creating a compute power grid," in *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 35–36, doi: 10.1109/CCGRID.2001.923173.
[98] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience: Research Articles," *Concurr. Comput. Pract. Exp.*, vol. 17, no. 2–4, pp. 323–356, 2005, doi: 10.1002/CPE.V17:2/4.
[99] K. Ousterhout, P. Wendell, M. Zaharia, and I. Stoica, "Sparrow," in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles - SOSP '13*, 2013, pp. 69–84, doi: 10.1145/2517349.2522716.
[100] S. L. Olivier, A. K. Porterfield, K. B. Wheeler, M. Spiegel, and J. F. Prins, "OpenMP task scheduling strategies for multicore NUMA systems," *Int. J. High Perform. Comput. Appl.*, vol. 26, no. 2, pp. 110–124, May 2012, doi: 10.1177/1094342011434065.
[101] M. Frigo, C. E. Leiserson, K. H. Randall, M. Frigo, C. E. Leiserson, and K. H. Randall, "The implementation of the Cilk-5 multithreaded language," *ACM SIGPLAN Not.*, vol. 33, no. 5, pp. 212–223, May 1998, doi: 10.1145/277652.277725.
[102] K. Wang, X. Zhou, T. Li, D. Zhao, M. Lang, and I. Raicu, "Optimizing load balancing and data-locality with data-aware scheduling," in *2014 IEEE International Conference on Big Data (Big Data)*, Oct. 2014, pp. 119–128, doi: 10.1109/BigData.2014.7004220.
[103] Z. Falt, M. Kruliš, D. Bednárek, J. Yaghob, and F. Zavoral, "Locality Aware Task Scheduling in Parallel Data Stream Processing," Springer, Cham, 2015, pp. 331–342.

[104] A. Muddukrishna, P. A. Jonsson, and M. Brorsson, "Locality-Aware Task Scheduling and Data Distribution for OpenMP Programs on NUMA Systems and Manycore Processors," *Sci. Program.*, vol. 2015, pp. 1–16, Nov. 2015, doi: 10.1155/2015/981759.
[105] W. Wei Ding, M. Yuanrui Zhang, M. Kandemir, J. Srinivas, and P. Yedlapalli, "Locality-aware mapping and scheduling for multicores," in *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, Feb. 2013, pp. 1–12, doi: 10.1109/CGO.2013.6495009.
[106] J. Lifflander, S. Krishnamoorthy, and L. V. Kale, "Optimizing Data Locality for Fork/Join Programs Using Constrained Work Stealing," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2014, pp. 857–868, doi: 10.1109/SC.2014.75.
[107] L. Xue *et al.*, "Locality-Aware Distributed Loop Scheduling for Chip Multiprocessors," in *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07)*, 2007, pp. 251–258, doi: 10.1109/VLSID.2007.97.
[108] M. Isard *et al.*, "Dryad," in *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007 - EuroSys '07*, 2007, vol. 41, no. 3, p. 59, doi: 10.1145/1272996.1273005.
[109] J. Maglalang, S. Krishnamoorthy, and K. Agrawal, "Locality-Aware Dynamic Task Graph Scheduling," in *2017 46th International Conference on Parallel Processing (ICPP)*, Aug. 2017, pp. 70–80, doi: 10.1109/ICPP.2017.16.
[110]R. M. Yoo, C. J. Hughes, C. Kim, Y.-K. Chen, and C. Kozyrakis, "Locality-Aware Task Management for Unstructured Parallelism: A Quantitative Limit Study," 2013.
[111] J. Paudel, O. Tardieu, and J. N. Amaral, "On the Merits of Distributed Work-Stealing on Selective Locality-Aware Tasks," in *2013 42nd International Conference on Parallel Processing*, Oct. 2013, pp. 100–109, doi: 10.1109/ICPP.2013.19.
[112] J. Choi, T. Adufu, and Y. Kim, "Data-Locality Aware Scientific Workflow Scheduling Methods in HPC Cloud Environments," *Int. J. Parallel Program.*, vol. 45, no. 5, pp. 1128–1141, Oct. 2017, doi: 10.1007/s10766-016-0463-0.
[113] Y. Guo, "A Scalable Locality-aware Adaptive Work-stealingScheduler for Multi-core Task Parallelism," RICE, 2011.
[114] B. Hindman *et al.*, "Mesos: a platform for fine-grained resource sharing in the data center," *Proceedings of the 8th USENIX conference on Networked systems design and implementation*. USENIX Association, pp. 295–308, 2011.
[115] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles - SOSP '09*, 2009, p. 261, doi: 10.1145/1629575.1629601.
[116] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, pp. 103–111, 1990, doi: 10.1145/79173.79181.
[117] Thomas Cheatham, Amr Fahmyy, Dan C. Stefanescu, and Leslie G. Valiant, "Bulk Synchronous Parallel Computing-A Paradigm for transportable Software," 1995.
[118] G. Malewicz *et al.*, "Pregel," in *Proceedings of the 2010 international conference on Management of data - SIGMOD '10*, 2010, p. 135, doi: 10.1145/1807167.1807184.
[119] "Apache Hama - Big Data and High-Performance Computing," 2012. .
[120] "Giraph," 2017. .
[121] J. M. D. Hill *et al.*, "BSPlib: The BSP programming library," *Parallel Comput.*, vol. 24, no. 14, pp. 1947–1980, Dec. 1998, doi: 10.1016/S0167-8191(98)00093-3.
[122] "BSPonMPI," 2006. .
[123] A. N. Yzelman, R. H. Bisseling, D. Roose, and K. Meerbergen, "MulticoreBSP for C : A high-performance library for shared-memory parallel programming," *Int. J. Parallel Program.*, vol. 42, no. 4, pp. 619–642, Jan. 2050, doi: http://dx.doi.org/10.1007/s10766-013-0262-9.
[124] A. N. Yzelman and R. H. Bisseling, "An object-oriented bulk synchronous parallel library for multicore programming," *Concurr. Comput. Pract. Exp.*, vol. 24, no. 5, pp. 533–553, Apr. 2012, doi: 10.1002/cpe.1843.
[125] J. M. Abello and J. S. Vitter, *External memory algorithms : DIMACS Workshop External Memory and Visualization, May 20-22, 1998*. American Mathematical Society, 1999.
[126] M. Kwiatkowska and R. Mehmood, "Out-of-Core Solution of Large Linear Systems of Equations Arising from Stochastic Modelling," Springer, Berlin, Heidelberg, 2002, pp. 135–151.
[127] R. Mehmood, "Disk-based Techniques for Efficient Solution of Large Markov Chains," PhD Thesis, School of Computer Science, University of Birmingham, 2004.
[128] M. Jung *et al.*, "Exploring the future of out-of-core computing with compute-local non-volatile memory," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*, 2013, pp. 1–11, doi: 10.1145/2503210.2503261.
[129] R. Koller, L. Marmol, R. Rangaswami, S. Sundararaman, N. Talagala, and M. Zhao, "Write policies for host-side flash caches," *Proceedings of the 11th USENIX conference on File and Storage Technologies*. USENIX Association, pp. 45–58, 2013.
[130] M. Saxena, M. M. Swift, and Y. Zhang, "FlashTier," in *Proceedings of the 7th ACM european conference on Computer Systems - EuroSys '12*, 2012, p. 267, doi: 10.1145/2168836.2168863.
[131] S. Byan, J. Lentini, A. Madan, and L. Pabon, "Mercury: Host-side flash caching for the data center," in *012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, Apr. 2012, pp. 1–12, doi: 10.1109/MSST.2012.6232368.
[132] E. Saule, H. M. Aktulga, C. Yang, E. G. Ng, and Ü. V. Çatalyürek, "An Out-of-Core Task-based Middleware for Data-Intensive Scientific Computing," in *Handbook on Data Centers*, New York, NY: Springer New York, 2015, pp. 647–667.
[133] E. Rothberg and R. Schreiber, "Efficient Methods for Out-of-Core Sparse Cholesky Factorization," *SIAM J. Sci. Comput.*, vol. 21, no. 1, pp. 129–144, Jan. 1999, doi: 10.1137/S1064827597322975.

[134] Mandhapati P and Siddhartha Khaitan, "High Performance Computing Using out-of-core Sparse Direct Solvers," *World Acad. Sci. Eng. Technol.*, 2009.
[135] A. Geist and R. Lucas, "Whitepaper on the Major Computer Science Challenges at Exascale," 2009.
[136] B. V. Dibyendu Das, Nagarajan Kathiresan, Rajan Ravindran, "Process mapping parallel computing," Nov. 2011.
[137] J. Hursey, J. M. Squyres, and T. Dontje, "Locality-Aware Parallel Process Mapping for Multi-core HPC Systems," in *2011 IEEE International Conference on Cluster Computing*, Sep. 2011, pp. 527–531, doi: 10.1109/CLUSTER.2011.59.
[138] E. R. Rodrigues, F. L. Madruga, P. O. A. Navaux, and J. Panetta, "Multi-core aware process mapping and its impact on communication overhead of parallel applications," in *2009 IEEE Symposium on Computers and Communications*, Jul. 2009, pp. 811–817, doi: 10.1109/ISCC.2009.5202271.
[139] M. J. Rashti, J. Green, P. Balaji, A. Afsahi, and W. Gropp, "Multi-core and Network Aware MPI Topology Functions," Springer, Berlin, Heidelberg, 2011, pp. 50–60.
[140] J. Hestness, S. W. Keckler, and D. A. Wood, "A comparative analysis of microarchitecture effects on CPU and GPU memory system behavior," in *2014 IEEE International Symposium on Workload Characterization (IISWC)*, Oct. 2014, pp. 150–160, doi: 10.1109/IISWC.2014.6983054.
[141] H. Chen, W. Chen, J. Huang, B. Robert, and H. Kuhn, "MPIPP," in *Proceedings of the 20th annual international conference on Supercomputing - ICS '06*, 2006, p. 353, doi: 10.1145/1183401.1183451.
[142] J. Zhang, J. Zhai, W. Chen, and W. Zheng, "Process Mapping for MPI Collective Communications," Springer, Berlin, Heidelberg, 2009, pp. 81–92.
[143] L. L. Pilla *et al.*, "A topology-aware load balancing algorithm for clustered hierarchical multi-core machines," *Futur. Gener. Comput. Syst.*, vol. 30, pp. 191–201, Jan. 2014, doi: 10.1016/j.future.2013.06.023.
[144] G. Zarrinchian, M. Soryani, and M. Analoui, "A New Process Placement Algorithm in Multi-core Clusters Aimed to Reducing Network Interface Contention," Springer, Berlin, Heidelberg, 2012, pp. 1041–1050.
[145]G. Mercier and J. Clet-Ortega, "Towards an Efficient Process Placement Policy for MPI Applications in Multicore Environments," Springer, Berlin, Heidelberg, 2009, pp. 104–115.
[146] P. Balaji, R. Gupta, A. Vishnu, and P. Beckman, "Mapping communication layouts to network hardware characteristics on massive-scale blue gene systems," *Comput. Sci. - Res. Dev.*, vol. 26, no. 3–4, pp. 247–256, Jun. 2011, doi: 10.1007/s00450-011-0168-y.
[147] B. E. Smith and B. Bode, "Performance Effects of Node Mappings on the IBM BlueGene/L Machine," Springer, Berlin, Heidelberg, 2005, pp. 1005–1013.
[148] H. Yu, I. Chung, and J. Moreira, "Topology Mapping for Blue Gene/L Supercomputer," in *ACM/IEEE SC 2006 Conference (SC'06)*, Nov. 2006, pp. 52–52, doi: 10.1109/SC.2006.63.
[149] S. Ito, K. Goto, and K. Ono, "Automatically optimized core mapping to subdomains of domain decomposition method on multicore parallel environments," *Comput. Fluids*, vol. 80, no. 1, pp. 88–93, Jul. 2013, doi: 10.1016/j.compfluid.2012.04.024.
[150] J. L. Traff, "Implementing the MPI Process Topology Mechanism," in *ACM/IEEE SC 2002 Conference (SC'02)*, 2002, pp. 28–28, doi: 10.1109/SC.2002.10045.
[151] J. Dümmler, T. Rauber, and G. Rünger, "Mapping Algorithms for Multiprocessor Tasks on Multi-Core Clusters," in *2008 37th International Conference on Parallel Processing*, Sep. 2008, pp. 141–148, doi: 10.1109/ICPP.2008.42.
[152] T. Hoefler and M. Snir, "Generic topology mapping strategies for large-scale parallel architectures," in *Proceedings of the international conference on Supercomputing - ICS '11*, 2011, p. 75, doi: 10.1145/1995896.1995909.
[153] L. V. Kale and S. Krishnan, "CHARM++: A portable concurrent object oriented system based on C++." University of Illinois at Urbana-Champaign, 1993.
[154] T. El-Ghazawi, *UPC : distributed shared memory programming*. Wiley, 2005.
[155] M. Castro, L. F. W. Goes, C. P. Ribeiro, M. Cole, M. Cintra, and J.-F. Mehaut, "A machine learning-based approach for thread mapping on transactional memory applications," in *2011 18th International Conference on High Performance Computing*, Dec. 2011, pp. 1–10, doi: 10.1109/HiPC.2011.6152736.
[156] D. Grewe and M. F. P. O'Boyle, "A Static Task Partitioning Approach for Heterogeneous Systems Using OpenCL," Springer, Berlin, Heidelberg, 2011, pp. 286–305.
[157] G. Tournavitis *et al.*, "Towards a holistic approach to auto-parallelization," *ACM SIGPLAN Not.*, vol. 44, no. 6, p. 177, May 2009, doi: 10.1145/1543135.1542496.
[158] Z. Wang, M. F. P. O'Boyle, Z. Wang, and M. F. P. O'Boyle, "Mapping parallelism to multi-cores," in *Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '09*, 2008, vol. 44, no. 4, p. 75, doi: 10.1145/1504176.1504189.
[159] S. Long, G. Fursin, and B. Franke, "A Cost-Aware Parallel Workload Allocation Approach Based on Machine Learning Techniques," in *Network and Parallel Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 506–515.
[160] F. Pinel, P. Bouvry, and B. Dorronsoro, "Savant: Automatic parallelization of a scheduling heuristic with machine learning," *Nat. Biol.*, 2013.
[161] M. K. Emani, M. O'Boyle, M. K. Emani, and M. O'Boyle, "Celebrating diversity: a mixture of experts approach for runtime mapping in dynamic environments," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation - PLDI 2015*, 2015, vol. 50, no. 6, pp. 499–508, doi: 10.1145/2737924.2737999.
[162] M. K. Emani and M. O'Boyle, "Change Detection Based Parallelism Mapping: Exploiting Offline Models and Online Adaptation," Springer International Publishing, 2015, pp. 208–223.

[163] C.-K. Luk, S. Hong, and H. Kim, "Qilin," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture - Micro-42*, 2009, p. 45, doi: 10.1145/1669112.1669121.
[164] J. González-Domínguez, G. L. Taboada, B. B. Fraguela, M. J. Martín, and J. Touriño, "Automatic mapping of parallel applications on multicore architectures using the Servet benchmark suite," *Comput. Electr. Eng.*, vol. 38, no. 2, pp. 258–269, Mar. 2012, doi: 10.1016/j.compeleceng.2011.12.007.
[165] D. Tiwari, S. S. Vazhkudai, Y. Kim, X. Ma, S. Boboila, and P. J. Desnoyers, "Reducing Data Movement Costs using Energy-Efficient, Active Computation on SSD."
[166] F. Zheng *et al.*, "GoldRush," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*, 2013, pp. 1–12, doi: 10.1145/2503210.2503279.
[167] C. Sewell *et al.*, "Large-scale compute-intensive analysis via a combined in-situ and co-scheduling workflow approach," *Proc. Int. Conf. High Perform. Comput. Networking, Storage Anal. - SC '15*, vol. 836, pp. 1–11, 2015, doi: 10.1145/2807591.2807663.
[168] S. Lakshminarasimhan *et al.*, "Compressing the Incompressible with ISABELA: In-situ Reduction of Spatio-temporal Data," Springer Berlin Heidelberg, 2011, pp. 366–379.
[169] H. Zou *et al.*, "Quality-Aware Data Management for Large Scale Scientific Applications," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, Nov. 2012, pp. 816–820, doi: 10.1109/SC.Companion.2012.114.
[170] J. Kim *et al.*, "Parallel in situ indexing for data-intensive computing," in *2011 IEEE Symposium on Large Data Analysis and Visualization*, Oct. 2011, pp. 65–72, doi: 10.1109/LDAV.2011.6092319.
[171] S. Lakshminarasimhan *et al.*, "Scalable in situ scientific data encoding for analytical query processing," *Proc. 22nd Int. Symp. High-performance parallel Distrib. Comput.*, pp. 1–12, 2013, doi: 10.1145/2462902.2465527.
[172] Y. Su, Y. Wang, and G. Agrawal, "In-Situ Bitmaps Generation and Efficient Data Analysis based on Bitmaps," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing - HPDC '15*, 2015, pp. 61–72, doi: 10.1145/2749246.2749268.
[173] H. Karimabadi, B. Loring, P. O'Leary, A. Majumdar, M. Tatineni, and B. Geveci, "In-situ visualization for global hybrid simulations," in *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment Gateway to Discovery - XSEDE '13*, 2013, p. 1, doi: 10.1145/2484762.2484822.
[174] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma, "In Situ Visualization for Large- Scale Combustion Simulations."
[175] H. Zou *et al.*, "FlexQuery: An online query system for interactive remote visual data exploration at large scale," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, Sep. 2013, pp. 1–8, doi: 10.1109/CLUSTER.2013.6702635.
[176] J. Woodring, J. Ahrens, T. J. Tautges, T. Peterka, V. Vishwanath, and B. Geveci, "On-demand unstructured mesh translation for reducing memory pressure during in situ analysis," in *Proceedings of the 8th International Workshop on Ultrascale Visualization - UltraVis '13*, 2013, pp. 1–8, doi: 10.1145/2535571.2535592.
[177] B. Nouanesengsy, J. Woodring, J. Patchett, K. Myers, and J. Ahrens, "ADR visualization: A generalized framework for ranking large-scale scientific data using Analysis-Driven Refinement," in *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*, Nov. 2014, pp. 43–50, doi: 10.1109/LDAV.2014.7013203.
[178] A. G. Landge *et al.*, "In-Situ Feature Extraction of Large Scale Combustion Simulations Using Segmented Merge Trees," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2014, pp. 1020–1031, doi: 10.1109/SC.2014.88.
[179] F. Zhang, S. Lasluisa, T. Jin, I. Rodero, H. Bui, and M. Parashar, "In-situ Feature-Based Objects Tracking for Large-Scale Scientific Simulations," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, Nov. 2012, pp. 736–740, doi: 10.1109/SC.Companion.2012.100.
[180] R. Mehmood, R. Meriton, G. Graham, P. Hennelly, and M. Kumar, "Exploring the influence of big data on city transport operations: a Markovian approach," *Int. J. Oper. Prod. Manag.*, vol. 37, no. 1, pp. 75–104, 2017, doi: 10.1108/IJOPM-03-2015-0179.
[181] R. Mehmood and G. Graham, "Big Data Logistics: A health-care Transport Capacity Sharing Model," *Procedia Comput. Sci.*, vol. 64, pp. 1107–1114, 2015, doi: 10.1016/j.procs.2015.08.566.
[182] E. Alomari;, I. Katib;, and R. Mehmood, "Iktishaf: A Big Data Road-Traffic Event Detection Tool Using Twitter and Spark Machine Learning," *Mob. Networks Appl.*, 2020, doi: 10.1007/s11036-020-01635-y.
[183] S. Alotaibi, R. Mehmood, I. Katib, O. Rana, and A. Albeshri, "Sehaa: A Big Data Analytics Tool for Healthcare Symptoms and Diseases Detection Using Twitter, Apache Spark, and Machine Learning," *Appl. Sci.*, vol. 10, no. 4, p. 1398, Feb. 2020, doi: 10.3390/app10041398.
[184] M. Aqib, R. Mehmood, A. Alzahrani, and I. Katib, *A smart disaster management system for future cities using deep learning, gpus, and in-memory computing*. 2020.
[185] M. Aqib, R. Mehmood, A. Alzahrani, I. Katib, A. Albeshri, and S. M. Altowaijri, "Rapid Transit Systems: Smarter Urban Planning Using Big Data, In-Memory Computing, Deep Learning, and GPUs," *Sustainability*, vol. 11, no. 10, p. 2736, May 2019, doi: 10.3390/su11102736.
[186] S. Suma, R. Mehmood, and A. Albeshri, "Automatic Detection and Validation of Smart City Events Using HPC and Apache Spark Platforms," in *Smart Infrastructure and Applications: Foundations for Smarter Cities and Societies*, Springer, 2020, pp. 55–78.
[187] S. Alotaibi and R. Mehmood, "Big data enabled healthcare supply chain management: Opportunities and challenges," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (LNICST), Volume 224*, Nov. 2018, vol. 224, pp. 207–215, doi: 10.1007/978-3-319-94180-6_21.

[188] I. Ahmad, F. Alqurashi, E. Abozinadah, and R. Mehmood, "Deep Journalism and DeepJournal V1.0: A Data-Driven Deep Learning Approach to Discover Parameters for Transportation," *Sustain.*, vol. 14, no. 9, p. 5711, May 2022, doi: 10.3390/SU14095711.
[189] Y. Arfat, S. Usman, R. Mehmood, and I. Katib, "Big data for smart infrastructure design: Opportunities and challenges," in *Smart Infrastructure and Applications Foundations for Smarter Cities and Societies*, Springer Cham, 2020, pp. 491–518.
[190] D. Singh *et al.*, "A survey on platforms for big data analytics," *J. Big Data*, vol. 2, no. 1, p. 8, Dec. 2015, doi: 10.1186/s40537-014-0008-6.
[191] J. Dean and S. Ghemawat, "MapReduce," *Commun. ACM*, vol. 51, no. 1, p. 107, Jan. 2008, doi: 10.1145/1327452.1327492.
[192] S. Ghemawat, H. Gobioff, S.-T. Leung, S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proceedings of the nineteenth ACM symposium on Operating systems principles - SOSP '03*, 2003, vol. 37, no. 5, p. 29, doi: 10.1145/945445.945450.
[193] T. White, *Hadoop : the Definitive Guide, 4th Edition*. .
[194] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, May 2010, pp. 1–10, doi: 10.1109/MSST.2010.5496972.
[195] D. Borthakur *et al.*, "Apache hadoop goes realtime at Facebook," in *Proceedings of the 2011 international conference on Management of data - SIGMOD '11*, 2011, p. 1071, doi: 10.1145/1989323.1989438.
[196] "Apache Tez," 2017. .
[197] J. Ekanayake *et al.*, "Twister," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing - HPDC '10*, 2010, p. 810, doi: 10.1145/1851476.1851593.
[198] R. P. Padhy, "Big Data Processing with Hadoop-MapReduce in Cloud Systems," *IJ-CLOSER - Int. J. Cloud Comput. Serv. Sci.*, vol. 2, no. 1, pp. 233–245, 2012.
[199] K. Singh and R. Kaur, "Hadoop: Addressing challenges of Big Data," in *2014 IEEE International Advance Computing Conference (IACC)*, Feb. 2014, pp. 686–689, doi: 10.1109/IAdCC.2014.6779407.
[200] H. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Map-reduce-merge," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data - SIGMOD '07*, 2007, p. 1029, doi: 10.1145/1247480.1247602.
[201] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and Good practices," *2013 6th Int. Conf. Contemp. Comput. IC3 2013*, pp. 404–409, 2013, doi: 10.1109/IC3.2013.6612229.
[202] R. Tudoran, A. Costan, and G. Antoniu, "MapIterativeReduce," in *Proceedings of third international workshop on MapReduce and its Applications Date - MapReduce '12*, 2012, p. 9, doi: 10.1145/2287016.2287019.
[203] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "HaLoop," *Proc. VLDB Endow.*, vol. 3, no. 1–2, pp. 285–296, Sep. 2010, doi: 10.14778/1920841.1920881.
[204] M. Zaharia *et al.*, "Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing," *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, pp. 2–2, 2012.
[205]C. L. Philip Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Inf. Sci. (Ny).*, vol. 275, pp. 314–347, 2014, doi: 10.1016/j.ins.2014.01.015.
[206] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins, "Pig latin," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, 2008, p. 1099, doi: 10.1145/1376616.1376726.
[207] Z. Lin, M. Cai, Z. Huang, and Y. Lai, "SALA: A Skew-Avoiding and Locality-Aware Algorithm for MapReduce-Based Join," Springer, Cham, 2015, pp. 311–323.
[208]S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, and L. Qi, "LEEN: Locality/Fairness-Aware Key Partitioning for MapReduce in the Cloud," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, Nov. 2010, pp. 17–24, doi: 10.1109/CloudCom.2010.25.
[209] R. Rhine and N. T. Bhuvan, "Locality Aware MapReduce," Springer, Cham, 2016, pp. 221–228.
[210] M. Y. Eltabakh, Y. Tian, F. Özcan, R. Gemulla, A. Krettek, and J. McPherson, "CoHadoop," *Proc. VLDB Endow.*, vol. 4, no. 9, pp. 575–585, Jun. 2011, doi: 10.14778/2002938.2002943.
[211] X. Yu and B. Hong, "Grouping Blocks for MapReduce Co-Locality," in *2015 IEEE International Parallel and Distributed Processing Symposium*, May 2015, pp. 271–280, doi: 10.1109/IPDPS.2015.16.
[212] J. Tan, S. Meng, X. Meng, and L. Zhang, "Improving ReduceTask data locality for sequential MapReduce jobs," in *2013 Proceedings IEEE INFOCOM*, Apr. 2013, pp. 1627–1635, doi: 10.1109/INFCOM.2013.6566959.
[213] J. Wang, Q. Xiao, J. Yin, and P. Shang, "DRAW: A New Data-gRouping-AWare Data Placement Scheme for Data Intensive Applications With Interest Locality," *IEEE Trans. Magn.*, vol. 49, no. 6, pp. 2514–2520, Jun. 2013, doi: 10.1109/TMAG.2013.2251613.
[214] Jiong Xie *et al.*, "Improving MapReduce performance through data placement in heterogeneous Hadoop clusters," in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, Apr. 2010, pp. 1–9, doi: 10.1109/IPDPSW.2010.5470880.
[215] R. M. Arasanal and D. U. Rumani, "Improving MapReduce Performance through Complexity and Performance Based Data Placement in Heterogeneous Hadoop Clusters," Springer, Berlin, Heidelberg, 2013, pp. 115–125.
[216] C.-W. Lee, K.-Y. Hsieh, S.-Y. Hsieh, and H.-C. Hsiao, "A Dynamic Data Placement Strategy for Hadoop in Heterogeneous Environments," *Big Data Res.*, vol. 1, pp. 14–22, Aug. 2014, doi: 10.1016/J.BDR.2014.07.002.
[217] V. Ubarhande, A.-M. Popescu, and H. Gonzalez-Velez, "Novel Data-Distribution Technique for Hadoop in Heterogeneous Cloud Environments," in *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, Jul. 2015, pp. 217–224, doi: 10.1109/CISIS.2015.37.

[218] S. Sujitha and S. Jaganathan, "Aggrandizing Hadoop in terms of node Heterogeneity & Data Locality," in *INTERNATIONAL CONFERENCE ON SMART STRUCTURES AND SYSTEMS - ICSSS'13*, Mar. 2013, pp. 145–151, doi: 10.1109/ICSSS.2013.6623017.
[219] Z. Guo, G. Fox, and M. Zhou, "Investigation of Data Locality in MapReduce," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, May 2012, pp. 419–426, doi: 10.1109/CCGrid.2012.42.
[220] Y. Chen, Z. Liu, T. Wang, and L. Wang, "Load Balancing in MapReduce Based on Data Locality," Springer, Cham, 2014, pp. 229–241.
[221] T.-Y. Chen, H.-W. Wei, M.-F. Wei, Y.-J. Chen, T. Hsu, and W.-K. Shih, "LaSA: A locality-aware scheduling algorithm for Hadoop-MapReduce resource assignment," in *2013 International Conference on Collaboration Technologies and Systems (CTS)*, May 2013, pp. 342–346, doi: 10.1109/CTS.2013.6567252.
[222] J. Park, D. Lee, B. Kim, J. Huh, and S. Maeng, "Locality-aware dynamic VM reconfiguration on MapReduce clouds," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing - HPDC '12*, 2012, p. 27, doi: 10.1145/2287076.2287082.
[223] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling."
[224] X. Zhang, Y. Feng, S. Feng, J. Fan, and Z. Ming, "An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments," in *2011 International Conference on Cloud and Service Computing*, Dec. 2011, pp. 235–242, doi: 10.1109/CSC.2011.6138527.
[225] C.-H. Hsu, K. D. Slagter, and Y.-C. Chung, "Locality and loading aware virtual machine mapping techniques for optimizing communications in MapReduce applications," *Futur. Gener. Comput. Syst.*, vol. 53, no. C, pp. 43–54, Dec. 2015, doi: 10.1016/j.future.2015.04.006.
[226] R. Xue, S. Gao, L. Ao, and Z. Guan, "BOLAS: Bipartite-Graph Oriented Locality-Aware Scheduling for MapReduce Tasks," in *2015 14th International Symposium on Parallel and Distributed Computing*, Jun. 2015, pp. 37–45, doi: 10.1109/ISPDC.2015.12.
[227] G. S. Sadasivam and D. Selvaraj, "A novel parallel hybrid PSO-GA using MapReduce to schedule jobs in Hadoop data grids," in *2010 Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, Dec. 2010, pp. 377–382, doi: 10.1109/NABIC.2010.5716346.
[228] X. Zhang, Y. Wu, and C. Zhao, "MrHeter: improving MapReduce performance in heterogeneous environments," *Cluster Comput.*, vol. 19, no. 4, pp. 1691–1701, Dec. 2016, doi: 10.1007/s10586-016-0625-2.
[229] L. Guo, H. Sun, and Z. Luo, "A Data Distribution Aware Task Scheduling Strategy for MapReduce System," Springer, Berlin, Heidelberg, 2009, pp. 694–699.
[230] M. Hammoud and M. F. Sakr, "Locality-Aware Reduce Task Scheduling for MapReduce," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, Nov. 2011, pp. 570–576, doi: 10.1109/CloudCom.2011.87.
[231] F. Ahmad *et al.*, "Tarazu," in *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '12*, 2012, vol. 40, no. 1, p. 61, doi: 10.1145/2150976.2150984.
[232] K. A. Kumar, V. K. Konishetty, K. Voruganti, and G. V. P. Rao, "CASH," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics - ICACCI '12*, 2012, p. 52, doi: 10.1145/2345396.2345406.
[233] Y. Zhao, W. Wang, D. Meng, Y. Lv, S. Zhang, and J. Li, "TDWS: A Job Scheduling Algorithm Based on MapReduce," in *2012 IEEE Seventh International Conference on Networking, Architecture, and Storage*, Jun. 2012, pp. 313–319, doi: 10.1109/NAS.2012.50.
[234] M. Hammoud, M. S. Rehman, and M. F. Sakr, "Center-of-Gravity Reduce Task Scheduling to Lower MapReduce Network Traffic," in *2012 IEEE Fifth International Conference on Cloud Computing*, Jun. 2012, pp. 49–58, doi: 10.1109/CLOUD.2012.92.
[235] S. Ibrahim, H. Jin, L. Lu, B. He, G. Antoniu, and S. Wu, "Maestro: Replica-Aware Map Scheduling for MapReduce," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, May 2012, pp. 435–442, doi: 10.1109/CCGrid.2012.122.
[236] K. K. Sethi and D. Ramesh, "Delay Scheduling with Reduced Workload on JobTracker in Hadoop," Springer, Cham, 2016, pp. 371–381.
[237] Y. Yang, J. Xu, F. Wang, Z. Ma, J. Wang, and L. Li, "A MapReduce Task Scheduling Algorithm for Deadline-Constraint in Homogeneous Environment," in *2014 Second International Conference on Advanced Cloud and Big Data*, Nov. 2014, pp. 208–212, doi: 10.1109/CBD.2014.35.
[238] A. Bezerra, P. Hernández, A. Espinosa, and J. C. Moure, "Job scheduling for optimizing data locality in Hadoop clusters," in *Proceedings of the 20th European MPI Users' Group Meeting on - EuroMPI '13*, 2013, p. 271, doi: 10.1145/2488551.2488591.
[239] M. Sun, H. Zhuang, C. Li, K. Lu, and X. Zhou, "Scheduling algorithm based on prefetching in MapReduce clusters," *Appl. Soft Comput.*, vol. 38, no. C, pp. 1109–1118, Jan. 2016, doi: 10.1016/j.asoc.2015.04.039.
[240] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. USENIX Association, pp. 10–10, 2010.
[241] Ken Hess, "Hadoop vs. Spark: The New Age of Big Data," 2016. .
[242] bernard Marr, "Spark Or Hadoop -- Which Is The Best Big Data Framework?," 2015. .
[243] S. Li *et al.*, "Stark: Optimizing In-Memory Computing for Dynamic Dataset Collections," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 103–114, doi: 10.1109/ICDCS.2017.143.
[244] C. Engle *et al.*, "Shark," in *Proceedings of the 2012 international conference on Management of Data - SIGMOD '12*, 2012, p. 689, doi: 10.1145/2213836.2213934.

[245] E. Santos-Neto, W. Cirne, F. Brasileiro, and A. Lima, "Exploiting Replication and Data Reuse to Efficiently Schedule Data-Intensive Applications on Grids," Springer, Berlin, Heidelberg, 2005, pp. 210–232.
[246] R. S. Xin, J. E. Gonzalez, M. J. Franklin, and I. Stoica, "GraphX," in *First International Workshop on Graph Data Management Experiences and Systems - GRADES '13*, 2013, pp. 1–6, doi: 10.1145/2484425.2484427.
[247] J. Goldstein, R. Ramakrishnan, and U. Shaft, "Compressing relations and indexes," in *Proceedings 14th International Conference on Data Engineering*, pp. 370–379, doi: 10.1109/ICDE.1998.655800.
[248] J. R. Larus, M. D. Hill, and T. M. Chilimbi, "Making pointer-based data structures cache conscious," *Computer (Long. Beach. Calif).*, vol. 33, no. 12, pp. 67–74, 2000, doi: 10.1109/2.889095.
[249] D. J. Abadi, S. R. Madden, and N. Hachem, "Column-stores vs. row-stores," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, 2008, p. 967, doi: 10.1145/1376616.1376712.
[250] H. Plattner and Hasso, "A common database approach for OLTP and OLAP using an in-memory column database," in *Proceedings of the 35th SIGMOD international conference on Management of data - SIGMOD '09*, 2009, pp. 1–2, doi: 10.1145/1559845.1559846.
[251] G. P. Copeland, S. N. Khoshafian, G. P. Copeland, and S. N. Khoshafian, "A decomposition storage model," in *Proceedings of the 1985 ACM SIGMOD international conference on Management of data - SIGMOD '85*, 1985, vol. 14, no. 4, pp. 268–279, doi: 10.1145/318898.318923.
[252] C. Kim *et al.*, "Designing fast architecture-sensitive tree search on modern multicore/many-core processors," *ACM Trans. Database Syst.*, vol. 36, no. 4, pp. 1–34, Dec. 2011, doi: 10.1145/2043652.2043655.
[253] V. Leis, A. Kemper, and T. Neumann, "The adaptive radix tree: ARTful indexing for main-memory databases," in *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, Apr. 2013, pp. 38–49, doi: 10.1109/ICDE.2013.6544812.
[254] L. M. Maas, T. Kissinger, D. Habich, and W. Lehner, "BUZZARD," in *Proceedings of the 2013 international conference on Management of data - SIGMOD '13*, 2013, p. 1285, doi: 10.1145/2463676.2465342.
[255] M.-C. Albutiu, A. Kemper, and T. Neumann, "Massively Parallel Sort-Merge Joins in Main Memory Multi-Core Database Systems," Jun. 2012.
[256] V. Leis, P. Boncz, A. Kemper, and T. Neumann, "Morsel-driven parallelism," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD '14*, 2014, pp. 743–754, doi: 10.1145/2588555.2610507.
[257] Y. Li, I. Pandis, R. Mueller, V. Raman, and G. Lohman, "NUMA-aware algorithms: the case of data shuffling," 2013.
[258] G. W. Burr *et al.*, "Phase change memory technology," *J. Vac. Sci. Technol. B, Nanotechnol. Microelectron. Mater. Process. Meas. Phenom.*, vol. 28, no. 2, pp. 223–262, Mar. 2010, doi: 10.1116/1.3301579.
[259] J. J. Yang and R. S. Williams, "Memristive devices in computing system," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, pp. 1–20, May 2013, doi: 10.1145/2463585.2463587.
[260] D. Apalkov *et al.*, "Spin-transfer torque magnetic random access memory (STT-MRAM)," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 9, no. 2, pp. 1–35, May 2013, doi: 10.1145/2463585.2463589.
[261] X. Shi *et al.*, "Mammoth: Gearing Hadoop Towards Memory-Intensive MapReduce Applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 8, pp. 2300–2315, Aug. 2015, doi: 10.1109/TPDS.2014.2345068.
[262] R. Power and J. Li, "Piccolo: building fast, distributed programs with partitioned tables," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, 2010, pp. 293–306.
[263] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed Stream Computing Platform," in *2010 IEEE International Conference on Data Mining Workshops*, Dec. 2010, pp. 170–177, doi: 10.1109/ICDMW.2010.172.
[264] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online," *Proceedings of the 7th USENIX conference on Networked systems design and implementation*. USENIX Association, pp. 21–21, 2010.
[265] V. Sikka, F. Färber, A. Goel, and W. Lehner, "SAP HANA," *Proc. VLDB Endow.*, vol. 6, no. 11, pp. 1184–1185, Aug. 2013, doi: 10.14778/2536222.2536251.
[266] T. Lahiri, M.-A. Neimat, and S. Folkman, "Oracle TimesTen: An In-Memory Database for Enterprise Applications."
[267] J. Lindström, J. Lindström, V. Raatikka, J. Ruuth, P. Soini, and K. Vakkila, "IBM solidDB: In-Memory Database Optimized for Extreme Speed and Availability."
[268] V. Raman *et al.*, "DB2 with BLU acceleration," *Proc. VLDB Endow.*, vol. 6, no. 11, pp. 1080–1091, Aug. 2013, doi: 10.14778/2536222.2536233.
[269] H. Zhang, G. Chen, B. C. Ooi, W.-F. Wong, S. Wu, and Y. Xia, "Anti-Caching”-based elastic memory management for Big Data," in *2015 IEEE 31st International Conference on Data Engineering*, Apr. 2015, pp. 1268–1279, doi: 10.1109/ICDE.2015.7113375.
[270] R. Gandhi, A. Gupta, A. Povzner, W. Belluomini, and T. Kaldewey, "Mercury," in *Proceedings of the 6th International Systems and Storage Conference on - SYSTOR '13*, 2013, p. 1, doi: 10.1145/2485732.2485746.
[271] B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, and R. Velkov, "OWLIM: A family of scalable semantic repositories," *Semant. Web*, vol. 2, no. 1, pp. 33–42, Jan. 2011, doi: 10.3233/SW-2011-0026.
[272] "memcached - a distributed memory object caching system." .
[273] G. Ananthanarayanan *et al.*, "PACMan: coordinated memory caching for parallel jobs," *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, pp. 20–20, 2012.
[274] F. Chang *et al.*, "Bigtable," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–26, Jun. 2008, doi: 10.1145/1365815.1365816.
[275] J. Martinec, A. Rango, and E. Major, "The Snowmelt-Runoff Model (SRM) user's manual," Apr. 1983.

[276] A. Rajasekar *et al.*, "iRODS Primer: Integrated Rule-Oriented Data System," *Synth. Lect. Inf. Concepts, Retrieval, Serv.*, vol. 2, no. 1, pp. 1–143, Jan. 2010, doi: 10.2200/S00233ED1V01Y200912ICR012.
[277] S. J. Plimpton and K. D. Devine, "MapReduce in MPI for Large-scale graph algorithms," *Parallel Comput.*, vol. 37, no. 9, pp. 610–632, Sep. 2011, doi: 10.1016/j.parco.2011.02.004.
[278] P. K. Mantha, A. Luckow, and S. Jha, "Pilot-MapReduce," in *Proceedings of third international workshop on MapReduce and its Applications Date - MapReduce '12*, 2012, p. 17, doi: 10.1145/2287016.2287020.
[279] P. Schwan and P. Schwan, "Lustre: Building a file system for 1000-node clusters," *PROC. 2003 LINUX Symp.*, 2003.
[280] S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Calvert, "PVS System Guide," 2001.
[281] E. Jeannot, G. Mercier, and F. Tessier, "Process Placement in Multicore Clusters:Algorithmic Issues and Practical Techniques," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 4, pp. 993–1002, Apr. 2014, doi: 10.1109/TPDS.2013.104.
[282] Y. Wang, "Smart : A MapReduce-Like Framework for In-Situ Scientific Analytics," 2015.
[283] C. Xu, R. Goldstone, Z. Liu, H. Chen, B. Neitzel, and W. Yu, "Exploiting Analytics Shipping with Virtualized MapReduce on HPC Backend Storage Servers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 1, pp. 185–196, Jan. 2016, doi: 10.1109/TPDS.2015.2389262.
[284] OLCF Staff Writer, "OLCF Group to Offer Spark On-Demand Data Analysis," 2016. .
[285] "Apache Hadoop 2.9.0 – C API libhdfs," 2017. .
[286] H. Jin, J. Ji, X.-H. Sun, Y. Chen, and R. Thakur, "CHAIO: Enabling HPC Applications on Data-Intensive File Systems," in *2012 41st International Conference on Parallel Processing*, Sep. 2012, pp. 369–378, doi: 10.1109/ICPP.2012.1.
[287] T. Hoefler, A. Lumsdaine, and J. Dongarra, "Towards Efficient MapReduce Using MPI."
[288] A. Matsunaga, M. Tsugawa, and J. Fortes, "CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications," in *2008 IEEE Fourth International Conference on eScience*, Dec. 2008, pp. 222–229, doi: 10.1109/eScience.2008.62.
[289] "HTCondor - High Throughput Computing," 2017. .
[290] Z. Zhang *et al.*, "Scientific computing meets big data technology: An astronomy use case," in *2015 IEEE International Conference on Big Data (Big Data)*, Oct. 2015, pp. 918–927, doi: 10.1109/BigData.2015.7363840.
[291] X. Lu, B. Wang, L. Zha, and Z. Xu, "Can MPI Benefit Hadoop and MapReduce Applications?," in *2011 40th International Conference on Parallel Processing Workshops*, Sep. 2011, pp. 371–379, doi: 10.1109/ICPPW.2011.56.
[292] J. Veiga, R. R. Exp, G. L. Taboada, and J. Touri, "Analysis and Evaluation of Big Data Computing Solutions in an HPC Environment," 2015.
[293] H. Mohamed and S. Marchand-Maillet, "Enhancing MapReduce Using MPI and an Optimized Data Exchange Policy," in *2012 41st International Conference on Parallel Processing Workshops*, Sep. 2012, pp. 11–18, doi: 10.1109/ICPPW.2012.6.
[294] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, "Evaluating MapReduce for Multi-core and Multiprocessor Systems," in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, 2007, pp. 13–24, doi: 10.1109/HPCA.2007.346181.
[295] X. Lu, M. W. U. Rahman, N. Islam, D. Shankar, and D. K. Panda, "Accelerating Spark with RDMA for Big Data Processing: Early Experiences," in *2014 IEEE 22nd Annual Symposium on High-Performance Interconnects*, Aug. 2014, pp. 9–16, doi: 10.1109/HOTI.2014.15.
[296] X. Lu, F. Liang, B. Wang, L. Zha, and Z. Xu, "DataMPI: Extending MPI to Hadoop-like Big Data Computing*," doi: 10.1109/IPDPS.2014.90.
[297] Y. Wang *et al.*, "Assessing the Performance Impact of High-Speed Interconnects on MapReduce," pp. 148–163, 2014, doi: 10.1007/978-3-642-53974-9_13.
[298] W. Weikuan Yu, Y. Yandong Wang, and X. Xinyu Que, "Design and Evaluation of Network-Levitated Merge for Hadoop Acceleration," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 602–611, Mar. 2014, doi: 10.1109/TPDS.2013.59.
[299] Alex Woodie, "Does InfiniBand Have a Future on Hadoop?," *HPC wire*, 2015. .
[300] "Unstructured Data Accelerator (UDA)," 2013. .
[301] "Mellanox Technologies: End-to-End InfiniBand and Ethernet Interconnect Solutions and Services." .
[302] V. K. J. Chu, "Transmission of IP over InfiniBand (IPoIB)," 2006.
[303] Alex Woodie, "Unravelling Hadoop Performance Mysteries," 2014. .
[304] N. S. Islam, X. Lu, M. Wasi-ur-Rahman, and D. K. Panda, "Can Parallel Replication Benefit Hadoop Distributed File System for High Performance Interconnects?," in *2013 IEEE 21st Annual Symposium on High-Performance Interconnects*, Aug. 2013, pp. 75–78, doi: 10.1109/HOTI.2013.24.
[305] M. Wasi-ur-Rahman *et al.*, "High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand," in *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, May 2013, pp. 1908–1917, doi: 10.1109/IPDPSW.2013.238.
[306] N. S. Islam *et al.*, "High performance RDMA-based design of HDFS over InfiniBand," in *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2012, pp. 1–12, doi: 10.1109/SC.2012.65.
[307] X. Lu *et al.*, "High-Performance Design of Hadoop RPC with RDMA over InfiniBand," in *2013 42nd International Conference on Parallel Processing*, Oct. 2013, pp. 641–650, doi: 10.1109/ICPP.2013.78.
[308] M. Turilli, M. Santcroos, and S. Jha, "A Comprehensive Perspective on Pilot-Job Systems," Aug. 2015.
[309] M. 122. Jones, M, Nelson, "Moving ahead with Hadoop YARN," 2016. .

[310] D. Petcu *et al.*, "On Processing Extreme Data," *Scalable Comput. Pract. Exp.*, vol. 16, no. 4, pp. 467–490, Jan. 2016, doi: 10.12694/scpe.v16i4.1134.
[311] G. Da Costa *et al.*, "Exascale Machines Require New Programming Paradigms and Runtimes," *Supercomput. Front. Innov.*, vol. 2, no. 2, pp. 6–27, Apr. 2015, doi: 10.14529/jsfi150201.
[312] S. Usman, R. Mehmood, I. Katib, A. Albeshri, and S. M. Altowaijri, "ZAKI: A Smart Method and Tool for Automatic Performance Optimization of Parallel SpMV Computations on Distributed Memory Machines," *Mob. Networks Appl.*, 2019.
[313] S. Usman, R. Mehmood, I. Katib, and A. Albeshri, "ZAKI+: A Machine Learning Based Process Mapping Tool for SpMV Computations on Distributed Memory Architectures," *IEEE Access*, vol. 7, pp. 81279–81296, 2019, doi: 10.1109/ACCESS.2019.2923565.
[314] M. K. Emani, M. F. P. Zheng Wang, and M. F. P. O'Boyle, "Smart, adaptive mapping of parallelism in the presence of external workload," in *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, Feb. 2013, pp. 1–10, doi: 10.1109/CGO.2013.6495010.
[315] Matthias Diener, "Automatic task and data mapping in shared memory architectures," Technische Universität Berlin, 2015.
[316] H. Subramoni, "Topology-Aware MPI Communication and Scheduling for High Performance Computing Systems," Ohio State University: Computer Science and Engineering., 2013.
[317] M. Kulkarni *et al.*, "Optimistic parallelism requires abstractions," in *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation - PLDI '07*, 2007, vol. 42, no. 6, p. 211, doi: 10.1145/1250734.1250759.
[318] M. (Intel) Keutzer, Kurt (EECS UC Berkeley), Tim, "Our Pattern Language _ Our Pattern Language." WordPress, 2016.
[319] S. J. Divakar Mysore with Shrikant Khupat, "Big data architecture and patterns, Part 1: Introduction to big data classification and architecture," *IBM*, 2013. .
[320] M. Zanoni, F. Arcelli Fontana, and F. Stella, "On applying machine learning techniques for design pattern detection," *J. Syst. Softw.*, vol. 103, pp. 102–117, May 2015, doi: 10.1016/J.JSS.2015.01.037.
[321] A. K. Dwivedi, A. Tirkey, R. B. Ray, and S. K. Rath, "Software design pattern recognition using machine learning techniques," in *2016 IEEE Region 10 Conference (TENCON)*, Nov. 2016, pp. 222–227, doi: 10.1109/TENCON.2016.7847994.