iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 4 **Issue:** VII **Month of publication:** July 2016

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A Study On Near Data Processing

Rucha Shankar Jamale
Department of Computer Engineering,
Bharati Vidyapeeth Deemed Universiy, Pune, India.

Abstract-This paper helps to understand the basics of Near Data Processing. Systems accessing big data need huge memory storage which results in quality less performance and high cost hence the term Near Data Processing emerged. In NDP computations are done within the same memory block where the respective data resides which helps to reduce execution time, memory consumption, leading to a seamless performance.
Keywords- Big Data, Active Storage, Near Data Processing, Disk Performance, FPGA, SCSI, Memory Model, Database Models.

I. INTRODUCTION

Systems which execute big data workloads usually move large data volumes from storage and memory for performing a reserved quantity of computation on each element of data. Data movement consumes high cost which results in limited performance and efficiency. To overcome this bottleneck and control the high bandwidths in memory and storage devices, a new idea evolved which includes computing near the data.

NDP minimizes the movement of data by computing at a suitable location in memory hierarchy; it considers the location and the information which is required to extract from the data. Hence computation is done exactly in data's residence, which is in main memory, cache or in persistent storage. This movement of data is done traditionally in CPU where it initially resides.NDP examples can be seen in existing systems in which computations are held near to the disk. Data streams are preprocessed into the disks such that minimum number of datasets can be shifted to the other parts of the system for processing.

II. LITERATURE REVIEW

A. Near-Data Processing

A Near data processing is an approach in which reducing movement of data that can be accomplished by shifting the computation near and more close to the data. It is referred by locating memory close to the data by creating a specialized hardware which supports small set of capabilities for computation; this idea was already proposed decades ago in various forms.

NDP hardware is not seen widely in terms of commercial products. NDP was originally proposed by the help of scaling Dennard and Moore's Law which included high and steady gain in performance of CPU. Due to withdrawing outcomes of scaling in technology renaissance in NDP research is possible.

In earlier days NDP systems have been focusing only on universal purpose computation. In recent years efforts were taken to understand the improvement opportunities in performance and power. Specialized NDP hardware accelerators provide these improvement opportunities. And these accelerators are derived from a finite set of attributes; hence it can remove many overhead resources related with general processing.

Conceptually, Keeping resources near to the data where they are located and reorganizing the applications in a distributed computing infrastructure; is the very principle which is applied at different steps at memory hierarchy and storage hierarchy.

B. Near-Data Computation

Moving the computation towards the main processor of the system and the processors embedded in SSD's (Solid State Drive) devices can produce large enhancement s relating bandwidth for data centric calculations. But the calculation latent goes afar which includes saving the power by avoiding the data movement across Serial Advanced Technology Attachment (SATA) or Peripheral Component Interconnect (PCI) Express. SSD's processors are more efficient than those in host, relating of energy per process. SSD code runs in a free execution environment hence it is trustworthy and can be acceptable easily. The accessed data's latency within SSD is less comparatively from the host.

Improvement in Latency benefits high performance, but influencing the advantages leads in efficiency of energy gain and declined latency in terms of difficult operations, which requires dependent data access (for e.g. atomicity guarantees and enforcing ordering). The software which implements these types of specific application semantics makes unpretentious computational command, but

moving this in the storage devices may lead to uneven impact on system's performance.

A newly proposed atomic, multipart, write mechanism decreases latency in terms of operational updates by 65% and mostly diminishes overheads in bandwidth which are incurred by logging schemes. Likely implementing some portions in an SSD for a finite key value stored in the processor may increase throughput by 7.5-9.6 times. Further gains could be provided by leveraging execution. A newly proposed storage line which allows bypassing operating system for storage attributes which do not change metadata of the file system could give good performance when file system can allot updates for easy metadata for the running software in an SSD. For discovering the prospective of specific application implementation in a storage system, it is also possible to implement a prototype in SSD which accepts programmability as the important idea in an interface of storage.

SSD applications are downloaded by various different applications for adding original features and modifying device behavior. SSD semantics helps to exploit all the NDP advantages. The newly derived SSD functionality works in a seamless manner with host side application when flexible programmability is provided within SSD.

It is comparatively easy to transfer legacy program portions to SSD with minimum efforts. In an SSD application information related to data structure and file systems is embedded so that metadata updates can be take captured by the SSD. The possibility of errors is reduced since similar data structures and algorithms are used by SSD application as original code at host side, and this results good performance leading less traffic I/O and simpler host side code to the SSD.

C. NDP Communication & Memory Model

The standard NDP systems are analogous to primary NDP designs that target plain workloads, such as parallel map phase in MapReduce. Many live applications, such as deep learning and graph processing, it needs more complex communication between thousands of threads. It will lead to performance bottleneck and energy wastage during moving of data between memory stacks and host processors when relayed completely on host processors for managing all the communications. Additionally, the number of NDP cores will develop along with memory capacity in a particular time. To completely utilize the execution and memory parallelism, we need an NDP architecture which supports efficient communication among thousands of threads. For direct communication in NDP cores, we support direct communication within NDP cores and across the stacks because it simplifies the implementation of communication patterns in memory analytics workloads.

The physical interconnection in each stack consists of a 2D mesh network-on-chip (NoC) on the logic die which allows cores related with each vault for direct communication with another vault within the same stack. Sharing a single router at each vault is effective area wise and adequate in terms of throughput. The 2D mesh provides access to the external serial links that helps to connect stacks with each other and to the host processor. This interconnection is allowed by all the cores in the system, by NDP and the host, to handle all the memory stacks through an integrated physical address space. NDP core sends read/write access straightly to its local vault controller. Virtual memory helps to assure Data coherence. Remote accesses are more expensive with respect to latency and energy. Though, analytics workload operates mainly on local data and communicates at easily understood points. By carefully optimizing the work assignment and data partitioning, NDP cores mostly accesses memory locations in their very own local vaults.

D. NDP in Database Systems

With a raising database applications storing all or recent data in huge main memory systems, memory wall leads to initial bottleneck. Various operators in database, like aggregation, selection, and projection, generate nearly same output as that of input with slight change, forming them agreeable for data movement optimization. Executing the above-mentioned operators directly within a memory and transporting only the required data (i.e., columns, tuples, and aggregates) by the subsystem of memory sets CPU liberated to perform different tasks, decrease pollution of cache, and improves bus pressure of memory. Alternatively, an NDP operator who generates bigger results compares to input, such as joins, which cannot assure improvement in performance for all time.

Operator select have progressed majorly in the latest years by applying techniques like working in vectorization, multicores, and data compressed; which helps only on memory bound. Adversely, for select operators NDP solution designing allow avoiding changing the data entirely. It is exciting that the NDP capability to shrink the whole data quantity passed towards the CPU by the help of hierarchy of memory equivalents the advantages gained by storage in columnar. Selected NDP can additionally change the data by moving up the related tuples of columns related, while storage in columnar requires entire related columns which are proliferated through the memory hierarchy.

E. NDP Features

1) Technology- Following two reasons make NDP a stronger case; without the help of earlier disadvantages of merged logic memory fabrication 3D and 2.5D technologies enabled the combination of computing and data stores; at a low energy overhead higher bandwidth is enabled by data and close immediacy of computation.

2) Balance- Tight coupling in every NDP element results in eliminating the computation to data bottleneck, but this can lead communication bottleneck between NDP elements. Fortunately, this can be recovered with new technologies like die-stacking and system-on-chip (SoC) technologies which enable integration of network-on-chip, which is a capable network stack of software and potentially leading openings for customized NDP interconnect designs.

3) Capacity- Including NVM in NDP result larger capacity in devices and usually lower cost. The benefit of gaining high capacity per element in NDP is the diminution in system for a fixed storage size of dataset. It is a merit because systems having small size leads low rate of failure and decrease obstacle in parallelization software. Earlier designs in NDP were limited due to capacity of small devices which included inter-device data movement and fine-grained parallelism.

4) Necessity- NDP is a promising alternative due to its high efficiency and improving overheads in centric architectures. It mainly involves following basics: Cache hierarchy overhead and data movement are reduced when computation is moved close towards the data; bandwidth, data capacity and locality are been matched while computing, which needs enabling of memory to computing ratio; and further improved efficiency by specialized computation for data transformation.

5) Secure workloads- A feasible secure market is vital for new technology implementations. Ideal market for establishing NDP is presented by Co-designed big-data appliances. Indeed, Oracle's Exadata and IBM's Netezza are old marketing NDP products.

6) Heterogeneity- the NDP involves heterogeneity in terms of flexible supports workloads at ample range. This could lead to a hurdle, but current improvements for managing heterogeneity (as FPGA/SoC, big. LITTLE, GPU/APU) and in programming are already clearing the path towards NDP adoption.

7) Ecosystem- Training, prototypes, and tools are vital elements for NDP implementation by non skilled programmers. Software programming prototypes such as MapReduce, Open MP4.0, and Open CL with their hardware model with the companies like Samsung, Micron, Vinray, and Micron provide basic knowledgw for developing the NDP applications.

8) Interface- Memory and host are the attributes needed by the NDP, which is literally impossible considering today's standard of DDRx. The importance of server and memory interfaces based on desktop is going to change following two trends: server and desktop DRAM which is replaced rapidly by mobile DRAM as latest memory service, and proliferation of recent memory interfaces like LPDDRx, HBM, DDR4, HMC, and Wide I/O. Moreover, interfaces like preliminary HMC includes support of NDP such as host device decoupling and smart refresh. More protocols in NDP can be developed by utilizing the above novel services.

9) Software- Map Reduce, a distributed software framework have admired the idea of putting computation close towards the data and eased the curve of learning of NDP hardware Programming. This type of framework even handles hard NDP software issues like naming, scheduling, data layout, and tolerating the fault.

10) Hierarchy- NVM i.e. new nonvolatile memory which integrates memory performance with storage capacity enables an even memory hierarchy and storage hierarchy and self-composed NDP computing elements. Particularly this hierarchy diminishes bottleneck resulting NDP data on and off the main memory.

F. Present Work

Recent companies working and producing in NDP applications are Vinray Technology, Netezza, Micron, Convey Computer, EMU Technology (the Automata Processor and Hybrid Memory Cube), Adapteva, DSSD, and Oracle.

III. SYSTEM CHALLENGES

The NDP systems make use of coarse-grained address interleaving rather than fine-grained address interleaving. To understand the impact of this alteration, we run the SPEC CPU2006 benchmarks on the Conv-3D system with fine-grained and coarse-grained interleaving. All processing is done itself on the host processor cores. For the benchmark that caches practically well in the host LLC (perlbench, gcc, etc.), the impact is significantly negligible. Among the rigorous memory benchmarks (libquantum, mcf, etc.), coarse-grained interleaving leads an average slowdown of 10 %(20.7% maximum for GemsFDTD). In general, this performance loss is not minor but it is not massive either. Hence, we it is worth to use coarse-grained interleaving to gain large benefits from NDP for in-memory analytics, even if some host-side code undergoes small degradation. Nevertheless, adaptive interleaving

schemes would be studied further in future work.

IV. CONCLUSION

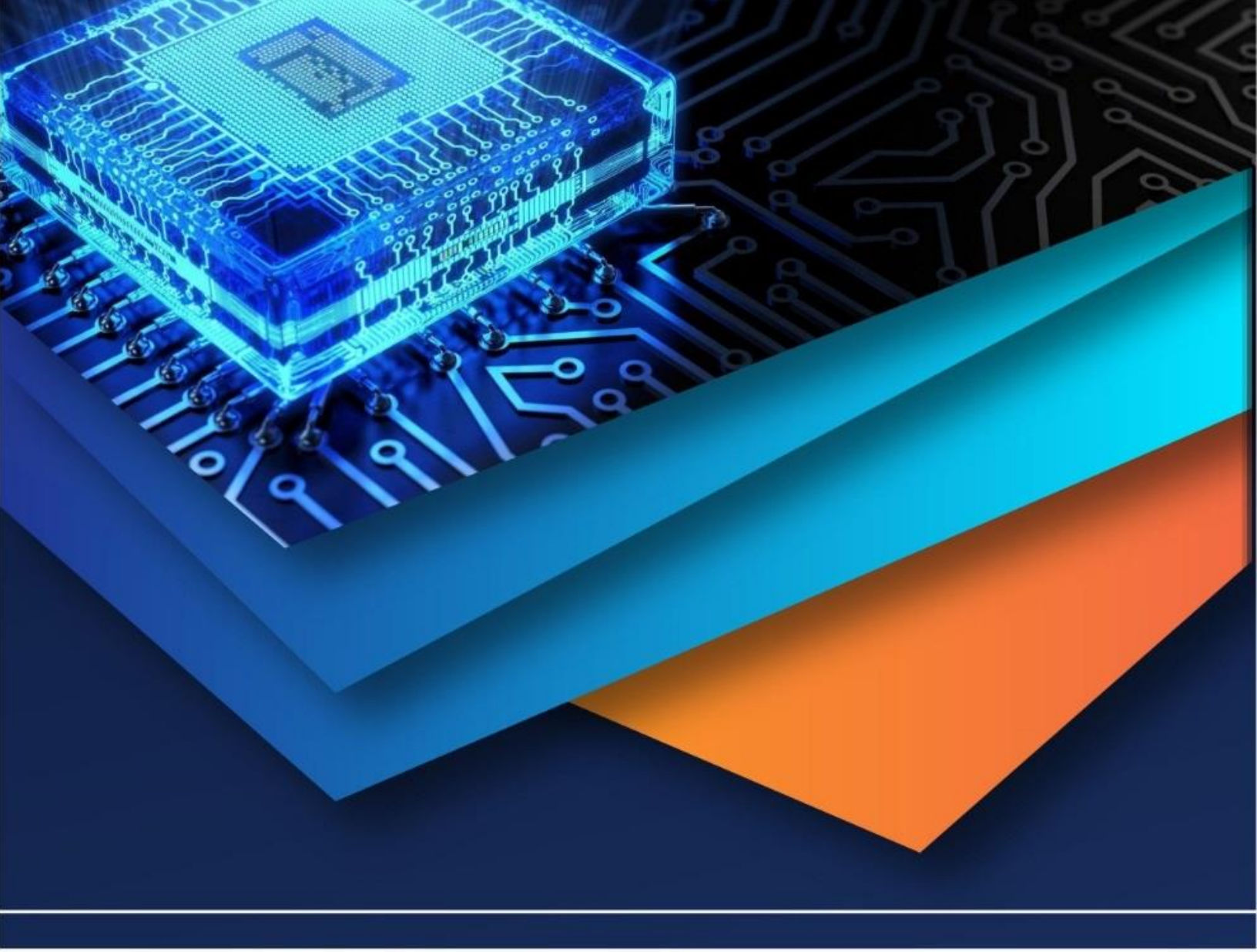
By placing data close to the memory, we reduce the energy waste for data movement in workloads having limited temporal locality. For better understanding we learned near data computation, near data communication, near data in databases and near data module.

REFERENCES

[1] Raval, K.S., Suryawanshi, R.S., Naveenkumar, J. and Thakore, D.M., 2011. The Anatomy of a Small-Scale Document Search Engine Tool: Incorporating a new Ranking Algorithm. International Journal of Engineering Science and Technology, 1(3), pp.5802-5808.
[2] Archana, R.C., Naveenkumar, J. and Patil, S.H., 2011. Iris Image Pre-Processing And Minutiae Points Extraction. International Journal of Computer Science and Information Security, 9(6), p.171.
[3] Jayakumar, M.N., Zaeimfar, M.F., Joshi, M.M. and Joshi, S.D., 2014. INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET). Journal Impact Factor, 5(1), pp.46-51.
[4] Naveenkumar, J. and Joshi, S.D., 2015. Evaluation of Active Storage System Realized through MobilityRPC.
[5] Jayakumar, D.T. and Naveenkumar, R., 2012. SDjoshi,“. International Journal of Advanced Research in Computer Science and Software Engineering,” Int. J, 2(9), pp.62-70.
[6] Jayakumar, N., Singh, S., Patil, S.H. and Joshi, S.D., Evaluation Parameters of Infrastructure Resources Required for Integrating Parallel Computing Algorithm and Distributed File System.
[7] Jayakumar, N., Bhardwaj, T., Pant, K., Joshi, S.D. and Patil, S.H., A Holistic Approach for Performance Analysis of Embedded Storage Array.
[8] Naveenkumar, J., Makwana, R., Joshi, S.D. and Thakore, D.M., 2015. OFFLOADING COMPRESSION AND DECOMPRESSION LOGIC CLOSER TO VIDEO FILES USING REMOTE PROCEDURE CALL. Journal Impact Factor, 6(3), pp.37-45.
[9] Naveenkumar, J., Makwana, R., Joshi, S.D. and Thakore, D.M., 2015. Performance Impact Analysis of Application Implemented on Active Storage Framework. International Journal, 5(2).
[10] Salunkhe, R., Kadam, A.D., Jayakumar, N. and Thakore, D., In Search of a Scalable File System State-of-the-art File Systems Review and Map view of new Scalable File system.
[11] Salunkhe, R., Kadam, A.D., Jayakumar, N. and Joshi, S., Luster A Scalable Architecture File System: A Research Implementation on Active Storage Array Framework with Luster file System.
[12] Jayakumar, N., Reducts and Discretization Concepts, tools for Predicting Student’s Performance.
[13] Jayakumar, M.N., Zaeimfar, M.F., Joshi, M.M. and Joshi, S.D., 2014. INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY (IJCET). Journal Impact Factor, 5(1), pp.46-51.
[14] Kumar, N., Angral, S. and Sharma, R., 2014. Integrating Intrusion Detection System with Network Monitoring. International Journal of Scientific and Research Publications, 4, pp.1-4.
[15] Namdeo, J. and Jayakumar, N., 2014. Predicting Students Performance Using Data Mining Technique with Rough Set Theory Concepts. International Journal, 2(2).
[16] Naveenkumar, J., Keyword Extraction through Applying Rules of Association and Threshold Values. International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), ISSN, pp.2278-1021.
[17] Kakamanshadi, G., Naveenkumar, J. and Patil, S.H., 2011. A Method to Find Shortest Reliable Path by Hardware Testing and Software Implementation. International Journal of Engineering Science and Technology (IJEST), ISSN, pp.0975-5462.
[18] Naveenkumar, J. and Raval, K.S., Clouds Explained Using Use-Case Scenarios.
[19] Naveenkumar J, S.D.J., 2015. Evaluation of Active Storage System Realized Through Hadoop. International Journal of Computer Science and Mobile Computing, 4(12), pp.67–73.
[20] RishikeshSalunkhe, N.J., 2016. Query Bound Application Offloading: Approach Towards Increase Performance of Big Data Computing. Journal of Emerging Technologies and Innovative Research, 3(6), pp.188–191.
[21] Sagar S lad s d joshi, N.J., 2015. Comparison study on Hadoop’s HDFS with Lustre File System. International Journal of Scientific Engineering and Applied Science, 1(8), pp.491–494.
[22] Salunkhe, R. et al., 2015. In Search of a Scalable File System State-of-the-art File Systems Review and Map view of new Scalable File system. In nternational Conference on electrical, Electronics, and Optimization Techni ques (ICEEOT) - 2016. pp. 1–8.
[23] BVDUCOE, B.B., 2011. Iris Image Pre-Processing and Minutiae Points Extraction. International Journal of Computer Science & Information Security.
[24] P. D. S. D. J. Naveenkumar J, “Evaluation of Active Storage System Realized through MobilityRPC,” Int. J. Innov. Res. Comput. Commun. Eng., vol. 3, no. 11, pp. 11329–11335, 2015
[25] N. Jayakumar, S. Singh, S. H. Patil, and S. D. Joshi, “Evaluation Parameters of Infrastructure Resources Required for Integrating Parallel Computing Algorithm and Distributed File System,” IJSTE, vol. 1, no. 12, pp. 251–254, 2015.
[26] N. Jayakumar, T. Bhardwaj, K. Pant, S. D. Joshi, and S. H. Patil, “A Holistic Approach for Performance Analysis of Embedded Storage Array,” Int. J. Sci. Technol. Eng., vol. 1, no. 12, pp. 247–250, 2015.
[27] J. Naveenkumar, R. Makwana, S. D. Joshi, and D. M. Thakore, “Performance Impact Analysis of Application Implemented on Active Storage Framework,” Int. J., vol. 5, no. 2, 2015.
[28] N. Jayakumar, “Reducts and Discretization Concepts, tools for Predicting Student’s Performance,” Int. J. Eng. Sci. Innov. Technol., vol. 3, no. 2, pp. 7–15, 2014.
[29] J. Namdeo and N. Jayakumar, “Predicting Students Performance Using Data Mining Technique with Rough Set Theory Concepts,” Int. J. Adv. Res. Comput. Sci. Manag. Stud., vol. 2, no. 2, 2014.
[30] R. Salunkhe, A. D. Kadam, N. Jayakumar, and S. Joshi, “Luster A Scalable Architecture File System: A Research Implementation on Active Storage Array

Framework with Luster file System.," in ICEEOT, 2015.

[31] Balasubramonian, R., Chang, J., Manning, T., Moreno, J.H., Murphy, R., Nair, R. and Swanson, S., 2014. Near-data processing: Insights from a MICRO-46 workshop. IEEE Micro, 34(4), pp.36-42.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY