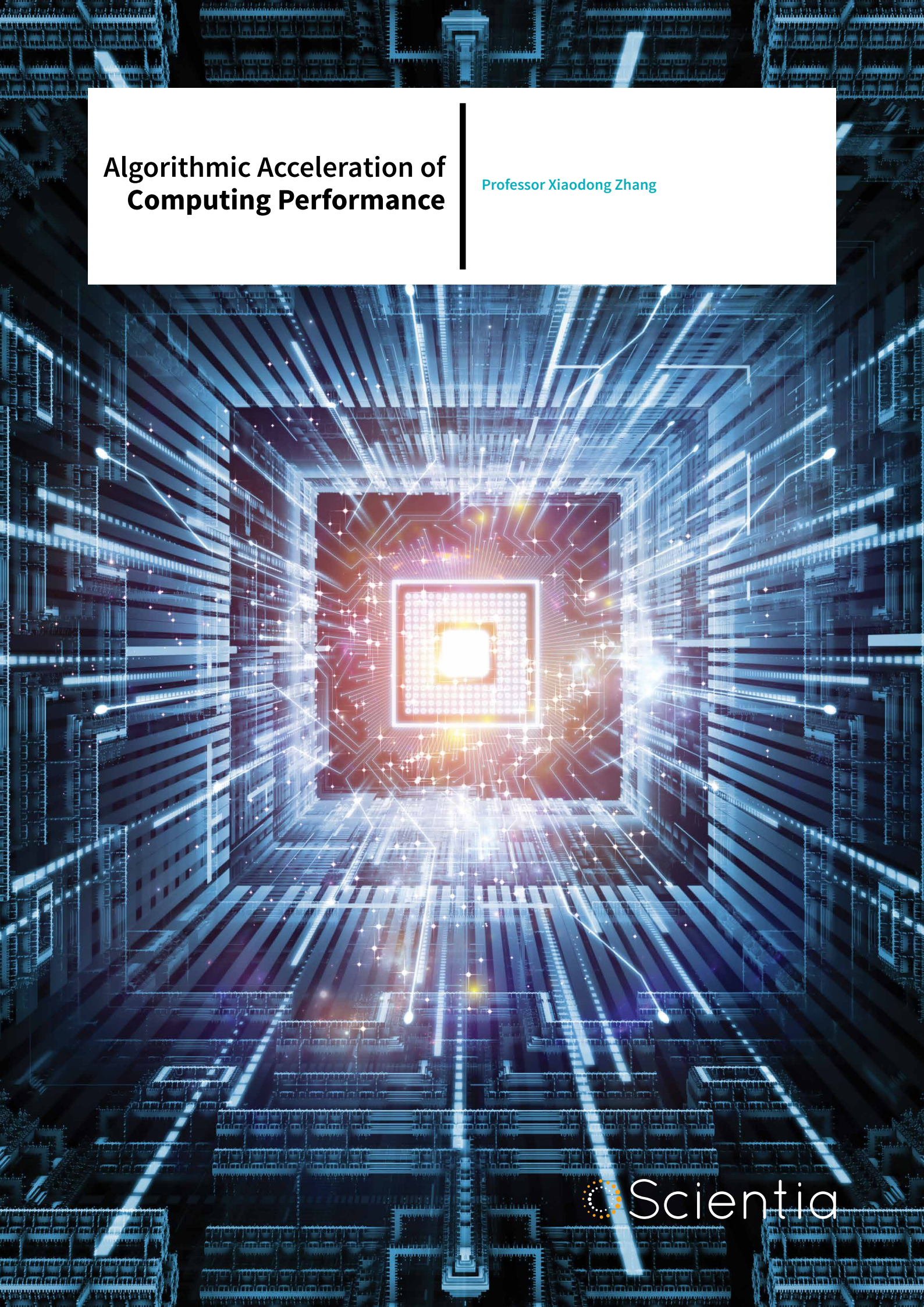
Algorithmic Acceleration of Computing Performance

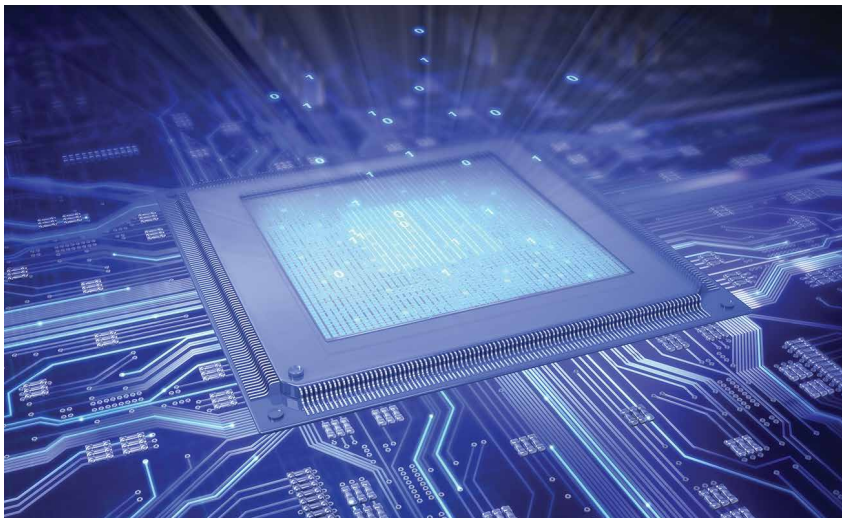
Professor Xiaodong Zhang



Scientia

ALGORITHMIC ACCELERATION OF COMPUTING PERFORMANCE

The endless quest for making faster, more powerful computers is not just about investing in advanced hardware. By developing more efficient algorithms, **Professor Xiaodong Zhang's** work has successfully revolutionised the design of fundamental computer components. By building upon basic computer science research on memory management processes, Professor Zhang has played a key role in improving overall computer system performance, not just for individual personal computers but also large computer clusters for data management systems.



Although loading a program to run on a computer might only require a few clicks from the user, there is a complex cascade of processes that are occurring while you wait. Your input from the mouse or a touch screen sends an electrical signal that is received by the computer's central processing unit (CPU). The CPU is like the computers 'brain', that allows it to respond to inputs, like mouse clicks, and executes the lines of code that make up a computer program. However, when you are trying to run a computer program and open its related data files, the code and data need to be stored somewhere, which is typically on the hard drive of the computer. This means that the CPU needs to access the data on the hard drive and find the correct program before it can even start running the program itself.

To run the program, a copy of the program and necessary data are first loaded into the main memory or DRAM (Dynamic Random Access Memory). Unlike a hard drive, where data is stored permanently, DRAM is a type of temporary storage. If you reboot your computer, then any data that is stored in the DRAM will be lost. The reason DRAM is used to store a copy of the program is because it is significantly faster to read from and write to DRAM compared with hard drive storage, ultimately meaning your program will run much more smoothly.

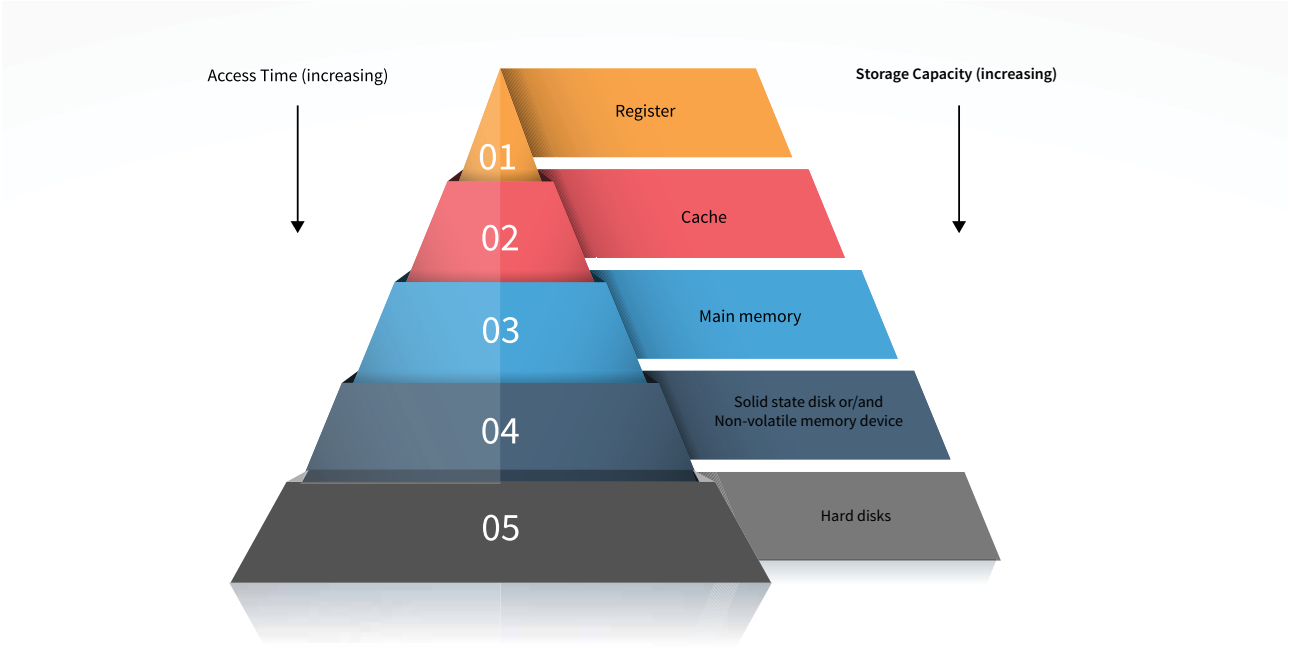
Every step in running a program requires communication between, and access to, several different kinds of memory in a hierarchy from fast to slow. Your operating system, be it Windows, Linux, OS X or Android, is continuously running on the CPU and helps to not only provide an interface for you to interact with but to translate various commands to and from the CPU and various parts of memory. It also tries to manage the overall memory usage of the computer.

Loading, Please Wait…

The continual 'back and forth' of commands between the CPU, hard disk and DRAM means that there are various stages at which 'bottlenecks' can occur, which limit the overall performance of a computer. For example, some types of computer programs, such as graphically-rich video games, are very 'DRAM intensive', so even with a significantly faster CPU, if your DRAM is insufficient, they may run sluggishly. While these bottlenecks may occur in individual components, a more significant and common source of bottlenecks is delays in the communication between the various kinds of memory.

One approach to solving this is to increase the memory capacity and upgrade CPU speed in the hardware. However, this is costly in terms of resources and it is becoming increasingly difficult to keep producing smaller and smaller transistors for CPUs, due to complicating factors such as the amount of heat produced. The most critical issue is to maximise every bit of potential of hardware by the power of algorithms. Where Professor Zhang and his research team excel is in finding different algorithms to optimise many of these 'memory management' processes. This ultimately means that you can see faster system performance using the same specification components by finding ways to make their operation or the communication between components more efficient.

‘Data accessing speed to the main memory is a critical factor for computer system performance. By translating basic research into advanced technologies, we have been able to have a huge impact of the development of the both the hardware and software components of computer systems.’



Reducing Conflict to Accelerate Memory Access

Every communication process, or attempt to read or write data from memory, has a certain probability of causing an error. There can also be inefficiencies caused by the order in which data is written to and read from memory.

For modern DRAM systems, one of the biggest inefficiencies is caused by row-buffer conflicts. In order to speed up memory access, the memory is divided into different ‘banks’, each of which contains multiple ‘pages’, a standard data unit, e.g. 8192 Bytes. Each bank has an associated row-buffer, which keeps data in memory so it can be re-accessed to speed up processing times.

However, sometimes an event called a ‘row buffer conflict’ occurs, which leads to a substantial delay in memory access. This is when there is an attempt to access a page while another page is already open in the row-buffer. To address this, Professor Zhang, with two of his former PhD students (Zhao Zhang and Zhichun Zhu, who are both now professors at the University of Illinois at Chicago) developed an algorithm known as ‘permutation-based page interleaving’ that significantly reduces the likelihood of these row-buffer conflict events.

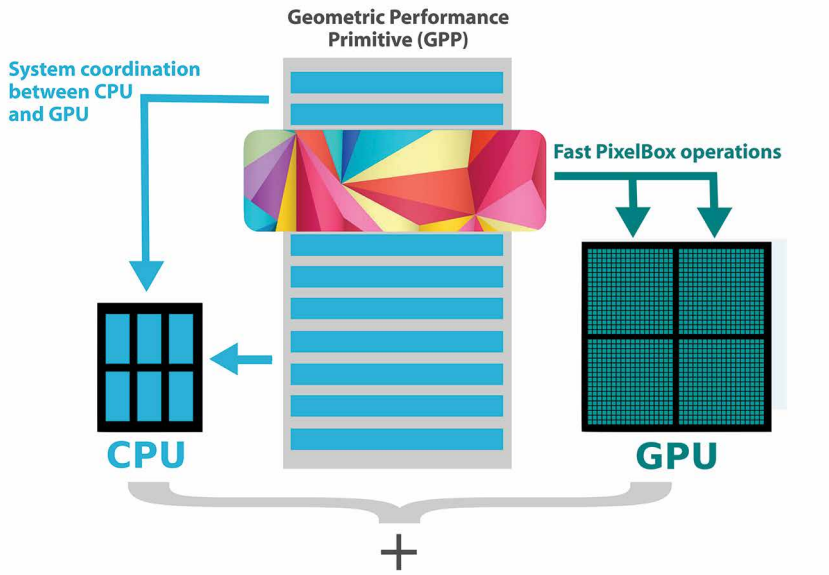
Use of Professor Zhang’s new algorithm has reduced memory access time significantly. This method has proved so useful that it has been adopted in almost all the general-purpose microprocessor products, originally by Sun Microsystems, and later by AMD, Intel and NVIDIA. It is now considered part of the conventional design of the memory controllers that make up the microprocessors to form the CPU.

To Keep or to Evict?

In any computer system, there are several layers of data buffers known as ‘caches’ in a hierarchy, used to store information the CPU is likely to need, reducing the time taken to access data from the next level of the memory hierarchy. For example, accessing data in CPU caches is much faster than in the main memory, and this trend continues until the last level of the hard disk is reached.

The question is, how do you identify what data is likely to be used often and what should be cached in order to speed up processing? This is normally done algorithmically, where the data is removed from the cache based on which data was ‘least recently used’ (LRU). However, now Professor Zhang, along with another former PhD student, Song Jiang (now a professor at the University of Texas, Arlington) have co-developed the Low Inter-reference Recency Set (LIRS) caching algorithm. This algorithm, and its approximation, have been widely adopted in major software products, including BSD and Linux operating systems, MySQL and H2 databases, Infinispan distributed systems, and other production systems.

Although LIRS may seem more complicated than the traditional LRU algorithm, it is a particularly effective method for data management in databases, operating systems, and data centres, because rather than just looking at when a specific page was accessed, it considers its reuse in a time interval, which is called ‘reuse distance’. Pages with short reuse distance are cached, improving the likelihood of caching the most useful data for improving performance.



Big Data Processing

'Big data' has been one of the most rapidly growing areas in society over recent years. Big data refers to such large and complex data sets that conventional data analysis tools simply cannot deal with them. However, despite these challenges, there has been such an explosion of interest in this field as it can offer important insights into a whole range of fields, including healthcare issues in the population and even predicting electoral outcomes.

Whereas standard PC hard drives often have around 1 Terabyte of storage space, this type of data set can be in excess of hundreds of terabytes. However, it is not just storing large volumes of data that poses a significant challenge – in order for this data to truly be useful, it needs to be rapidly accessible for analysis.

Conventional databases often store data in either a row or column store format. Row-store format has the advantage of being faster to load and includes a 'complete' dataset by rows in memory for analysis. However, not all the data elements in a row are used in practice, causing inefficient usage of limited storage bandwidth. Column-store means that any unnecessary or redundant data is not read into memory, but does not offer the same advantage of being a 'complete' data set. In addition, operations among distributed columns in different nodes require network communications.

Professor Zhang, in collaboration with research scientist Rubao Lee, former PhD students Yin Huai (now working at DataBricks) and Yuan Yuan (now working at Google), and several Facebook software engineers have designed and implemented a new type of data structure called Record Columnar File (RCFile) and its optimised version (ORC) that allows significantly faster memory access. With this hybrid structure containing both columns and rows, RCFIle and ORC retain the merits of both column-store and row-store methods, but minimise their limits. This has become the standard data storage format for large scale data processing systems, such as Hive (Hortenworks), Presto (Facebook), and Impala (Cloudera), but has also been adopted by major database vendors of IBM, Microsoft, Oracle, SAS, and Teradata for their commercial database products.

Algorithm Drives for Image and Graphics Processing

Generating computer graphics is typically one of the most demanding computational operations. When you are playing a video game, what you are actually seeing is shapes made up of polygons, rendered in different colours. As graphics have become increasingly realistic over the years, this means far higher polygon counts, which means more data to process and keep in memory.

Now, there is a great deal of interest in being able to visually represent the large datasets typical of the big data era. This makes use of the same polygon building blocks that make up video game graphics, but the size and complexity of what needs to be visualised often means that several layers of these polygons need to be used to reconstruct things like 3-D maps or automate diagnosis of medical conditions from images or scan data.

While graphical processing units (GPUs) are optimised for dealing with these types of processes, Professor Zhang, alongside former PhD students Kaibo Wang (now working at Google), Yin Huai, research scientist Rubao Lee, and two faculty members in Emory University, have found an algorithmic approach to greatly speed up these complex polygon overlay operations. This algorithm is known as PixelBox and has been adopted in the Geometric Performance Primitives (GPP) Library – an industry-leading and high speed computational geometry engine. GPP has also been included in the GPU-Accelerated libraries of the NVIDIA Company.

Future Directions

With data sets becoming ever larger, building powerful supercomputers for analysis is no longer an effective and affordable answer. Much of Professor Zhang's recent work involves 'distributed computing'. Rather than having one solitary powerful computer to run a big task, in a distributed computing arrangement, this task is split over multiple computers so the workload is shared and it is completed more efficiently in a cost-effective way.

However, there are significantly more memory management issues and other hardware challenges due to the rapid advancement of high performance devices, such as Graphics Processing Unit (GPU), Field-Programmable Gate Array (FPGA), Solid State Devices (SSD), and Non-Volatile Memory (NVM), when dealing with an entire network of computers rather than just one, and there is a need to design algorithms that exploit the unique capabilities of running distributed computing. Professor Zhang, alongside his group at The Ohio State University, will continue to spearhead the development of new algorithms and of an inclusive software environment for heterogeneous hardware devices to ensure this happens.

Meet the researcher

Professor Xiaodong Zhang
Computer Science and Engineering Department
Ohio State University
Columbus
USA

Professor Xiaodong Zhang graduated with a PhD in Computer Science from University of Colorado at Boulder, where he received a Distinguished Engineering Alumni Award in 2011. He is currently the Robert M. Critchfield Professor in Engineering and Chair of the Computer Science and Engineering Department at The Ohio State University. Professor Zhang has made significant contributions to the fields of computer memory systems and data and memory management in distributed systems, for which he was named as IEEE Fellow (Institute of Electronics and Electrical Engineers) in 2009 and ACM Fellow (Association for Computing Machinery) in 2012. His research focus has been on finding ways to translate basic computer science research into high-impact technologies, and much of this work has been contributed to the public domain through open-source software and published papers.

CONTACT

E: zhang@cse.ohio-state.edu
T: (+1) 614 292 2770
W: http://web.cse.ohio-state.edu/~zhang/

THE OHIO STATE UNIVERSITY

REFERENCES

Y Huai, S Ma, R Lee, O O'Malley, X Zhang, Understanding Insights into the Basic Structure and Essential Issues of Table Placement Methods in Clusters, Journal Proceedings of the VLDB Endowment, 2013, 6, 1750–1761.

K Wang, Y Huai, R Lee, F Wang, X Zhang, JH Saltz, Accelerating Pathology Image Data Cross-Comparison on CPU-GPU Hybrid Systems, Journal Proceedings of the VLDB Endowment, 2012, 5, 1543–1554.

Y He, R Lee, Y Huai, Z Shao, N Jain, X Zhang, Z Xu, RCFile: A Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse Systems, 2011 IEEE 27th International Conference on Data Engineering (ICDE), 2011, 1199–1208.

S Jiang, F Chen, X Zhang, CLOCK-Pro: An Effective Improvement of the CLOCK Replacement, Proceeding ATC'05 Proceedings of the annual conference on USENIX Annual Technical Conference, 2005, 35–55.

S Jiang, X Zhang, LIRS: An Efficient Low Inter-reference Recency Set Replacement Policy to Improve Buffer Cache Performance, SIGMETRICS '02 Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2002, 31–42.

Z Zhang, Z Zhu, X Zhang, A Permutation-based Page Interleaving scheme to Reduce Row-buffer Conflicts and Exploit Data Locality, Micro'33 Proceedings of the 33rd Annual ACM/IEEE International Symposium on Microarchitecture, 2000, 32–41.