Computer Architecture for Big Data

Behrooz Parhami
Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, USA

Synonyms

Big data hardware acceleration; Hardware considerations for big data

Definition

How features of general-purpose computer architecture impact big-data applications and, conversely, how requirements of big data lead to the emergence of new hardware and architectural support.

Overview

Computer architecture (Parhami 2005) is a subdiscipline of computer science and engineering that is concerned with designing computing structures to meet application requirements effectively, economically, reliably, and within prevailing technological constraints. In this entry, we discuss how features of general-purpose computer architecture impacts big-data applications and, conversely, how requirements of big data lead to the emergence of new hardware and architectural support.

Historical Trends in Computer Architecture

The von Neumann architecture for stored-program computers, with its single or unified memory, sometimes referred to as the Princeton architecture, emerged in 1945 (von Neumann 1945; von Neumann et al. 1947) and went virtually unchallenged for decades. It dominated the alternative Harvard architecture with separate program and data memories (Aiken and Hopper 1946) from the outset as the more efficient and versatile way of implementing digital computers.

As the workload for general-purpose computers began to change, adjustments in, and alternatives to, von Neumann architecture were proposed. Examples include de-emphasizing arithmetic operations in favor of data movement primitives, as seen in input/output and stream processors (Rixner 2001); introducing hardware aids for frequently used operations, as in graphic processing units or GPUs (Owens et al. 2008; Singer 2013); and adding special instructions for improved performance on multimedia workloads (Lee 1995; Yoon et al. 2001).

S. Sakr, A. Zomaya (eds.), *Encyclopedia of Big Data Technologies*,
https://doi.org/10.1007/978-3-319-63962-8_164-1

Recently, data-intensive applications necessitated another reassessment of the match between prevalent architectures and application requirements. The performance penalty of data having to be brought into the processor and sent back to memory through relatively narrow transfer channels was variously dubbed the "von Neumann bottleneck" (Markgraf 2007) and the "memory wall" (McKee 2004; Wulf and McKee 1995). Memory data transfer rates are measured in GB/s in modern machines, whereas the processing rates can be three or more decimal orders of magnitude higher.

The term "non-von" (Shaw 1982) was coined to characterize a large category of machines that relaxed one or more of the defining features of the von Neumann architecture, so as to alleviate some of the perceived problems. Use of cache memories (Smith 1982), often in multiple levels, eased the von Neumann bottleneck for a while, but the bottleneck reemerged, as the higher cache data transfer bandwidth became inadequate and applications that lacked or had relatively limited locality of reference emerged. Memory interleaving and memory-access pipelining, pioneered by IBM (Smotherman 2010) and later used extensively in Cray supercomputers, was the next logical step.

Extrapolating a bit from Fig. 1 (which covers the period 1985–2010), and using round numbers, the total effect of architectural innovations has been a 100-fold gain in performance, on top of another factor-of-100 improvement due to faster gates and circuits (Danowitz et al. 2012). Both growth rates in Fig. 1 show signs of slowing down, so that future gains to support the rising processing need of big data will have to come, at least in part, from other sources. In the technology arena, use of emerging technologies will provide some boost for specific applications. An intriguing option is resurrecting hybrid digital/analog computing, which was sidelined long ago in favor of all-digital systems. Architecturally, specialization is one possible avenue for maintaining the performance growth rate, as are massively parallel and in-memory or near-memory computing.

How Big Data Affects Computer Architecture

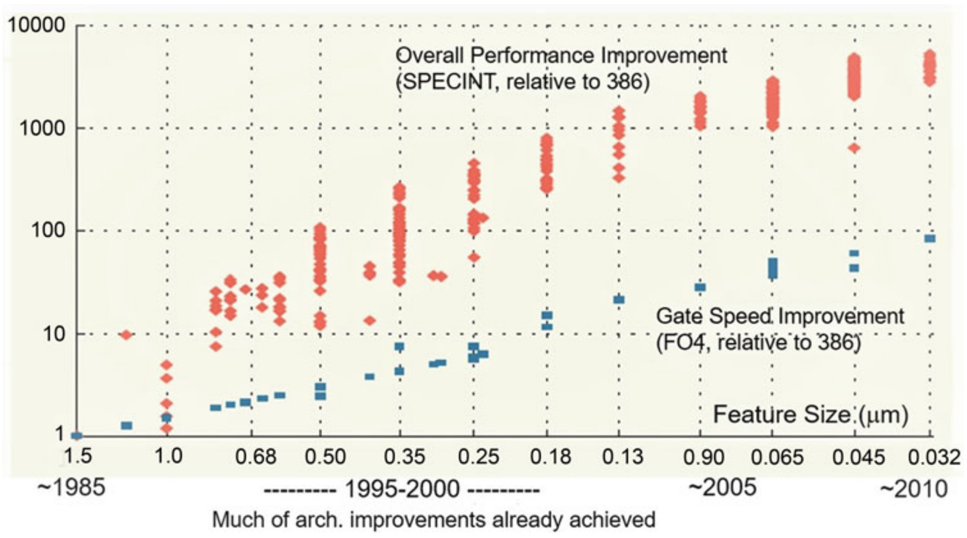
The current age of big data (Chen and Zhang 2014; Hu et al. 2014) has once again exposed the von Neumann bottleneck, forcing computer architects to seek new solutions to the age-old problem, which has become much more serious. Processing speed continues to rise exponentially, while memory bandwidth increases at a much slower pace.

It is by now understood that big data is different from "lots of data." It is sometimes defined in terms of the attributes of volume, variety, velocity, and veracity, known as the "4Vs" (or "5 Vs," if we also include value). Dealing with big data requires big storage, big-data processing capability, and big communication bandwidth. The first two (storage and data processing) directly affect the architecture of the nodes holding and processing the data. The part of communication that is internode is separately considered in this encyclopedia. However, there is also the issue of intranode communication represented in buses and networks-on-chip that belong to our architectural discussion here.

In addition to data volume, the type of data to be handled is also changing from structured data, as reflected, for example, in relational databases, to semi-structured and unstructured data. While this change has some negative effects in terms of making traditional and well-understood database technologies obsolete, it also opens up the possibility of using scalable processing platforms made of commodity hardware as part of the cloud-computing infrastructure. Massive unstructured data sets can be stored in distributed file systems, such as the ones designed in connection with Hadoop (Shafer et al. 2010) or SQL/noSQL (Cattell 2011).

In addition to the challenges associated with rising storage requirements and data access bandwidth, the processing load grows with data volume because of various needs. These are:

- Encryption and decryption
- Compression and decompression
- Sampling and summarization



Computer Architecture for Big Data, Fig. 1 Technology advances and architectural innovations each contributed a factor of ∼100 improvement in processor performance over three decades. (Danowitz et al. 2012)

- Visualization and graphing
- Sorting and searching
- Indexing and query processing
- Classification and data mining
- Deduction and learning

The first four items above, which are different forms of data translation, are discussed in the next section. The other items, viz., data transformations, will be discussed subsequently.

Architectural Aids to Data Translations

Many important data translations are handled by endowing a general-purpose architecture with suitable accelerator units deployed as coprocessors. Such accelerators are ideally custom integrated circuits, whose designs are fully optimized for their intended functions. However, in view of rapid advances in capabilities, performance, and energy efficiency of field-programmable gate arrays (Kuon et al. 2008), a vast majority of modern accelerators reported in the literature are built on FPGA circuits.

Accelerators for encryption and decryption algorithms have a long history (Bossuet et al. 2013). The binary choice of custom-designed VLSI or general-purpose processing for cryptographic computations has expanded to include a variety of intermediate solutions which include the use of FPGAs and GPUs. The best solution for an application domain depends not only on the required data rates and the crypto scheme, but also on power, area, and reliability requirements.

Data compression (Storer 1988) allows us to trade processing time and resources for savings in storage requirements. While any type of data can be compressed (e.g., text compression), massive sizes of video files make them a prime target for compression. With the emergence of video compression standards (Le Gall 1991), much effort has been expended to implement the standards on special-purpose hardware (Pirsch et al. 1995), offering orders of magnitude speed improvement over general-purpose programmed implementations.

Both sampling and summarization aim to reduce data volume while still allowing the operations of interest to be performed with reasonable precision. An alternative to post-collection reduction of data volume is to apply compression during data collection, an approach that in the case of sensor data collection is known as compressive sensing (Baraniuk 2007). Compressive sensing, when applicable, not only saves on processing time but also reduces transmission bandwidth and storage requirements. There are some mathematical underpinnings common to compressive sensing techniques, but at the implementation level, the methods and algorithms are by and large application-dependent.

Data visualization (Ward et al. 2010) refers to the production of graphical representation of data for better understanding of hidden structures and relationships. It may provide the only reasonable hope for understanding massive amounts of data, although machine learning is a complementary and competing method of late. Several visualization accelerators were implemented in the late 1990s (e.g., Scott et al. 1998), but the modern trend is to use FPGA and cluster-based methods.

Architectural Aids to Data Transformations

Sorting is an extremely important primitive that is time-consuming for large data sets. It is used in a wide array of contexts, which includes facilitating subsequent searching operations. It can be accelerated in a variety of ways, from building more efficient data paths and memory access schemes, in order to make conventional sorting algorithms run faster, to the extreme of using hardware sorting networks (Mueller et al. 2012; Parhami 1999).

Indexing is one of the most important operations for large data sets, such as those maintained and processed by Google. Indexing and query processing have been targeted for acceleration within large-scale database implementations (Casper and Olukotun 2014; Govindaraju et al. 2004). Given the dominance of relational databases in numerous application contexts, a variety of acceleration methods have been proposed for operations on such databases (e.g., Bandi et al. 2004). Hardware components used in realizing such accelerators include both FPGAs and GPUs.

Hardware aids for classification are as diverse as classification algorithms and their underlying applications. A prime example in Internet routing is packet classification (Taylor 2005), which is needed when various kinds of packets, arriving at extremely high rates, must be separated for appropriate handling. Modern hardware aids for packet classification use custom arrays for pipelined network processing, content-addressable memories (Liu et al. 2010), GPUs (Owens et al. 2008), or tensor processing units (Sato et al. 2017). Data mining, the process of generating new information by examining large data sets, has also been targeted for acceleration (Sklyarov et al. 2015), and it can benefit from similar highly parallel processing approaches. Also falling under such acceleration schemes are aids and accelerators for processing large graphs (Lee et al. 2017).

The earliest form of deduction engines were theorem provers. An important application of automatic deduction and proof is in hardware verification (Cyrluk et al. 1995). In recent years, machine learning has emerged as an important tool for improving the performance of conventional systems and for developing novel methods of tackling conceptually difficult problems. Game-playing systems (Chen 2016) constitute important testbeds for evaluating various approaches to machine learning and their associated hardware acceleration mechanisms. This is a field that has just started its meteoric rise and bears watching for future applications.

Memory, Processing, and Interconnects

In both general-purpose and special-purpose systems interacting with big data, the three interconnected challenges of providing adequate memory capacity, supplying the requisite processing power, and enabling high-bandwidth data movements between the various data-handling nodes must be tackled (Hilbert and Lopez 2011).

The memory problem can be approached using a combination of established and novel technologies, including nonvolatile RAM, 3D stacking of memory cells, processing in memory, content-addressable memory, and a variety of novel (nanoelectronics or biologically inspired) technologies. We won't dwell on the memory architecture in this entry, because the memory challenge is addressed in other articles (see the Cross-References).

Many established methods exist for increasing the data-handling capability of a processing node. The architectural nomenclature includes superscalar and VLIW organizations, collectively known as instruction-level parallelism (Rau and Fisher 1993), hardware multithreading (Eggers et al. 1997), multicore parallelism (Gepner and Kowalik 2006), domain-specific hardware accelerators (examples cited earlier in this entry), transactional memory (Herlihy and Moss 1993), and SIMD/vector architectural or instruction-set extensions (Lee 1995; Yoon et al. 2001). A complete discussion of all these methods is beyond the scope of this entry, but much pertinent information can be found elsewhere in this encyclopedia.

Intranode communication is achieved through high-bandwidth bus systems (Hall et al. 2000) and, increasingly, for multicore processors and systems-on-chip, by means of on-chip networks (Benini and De Micheli 2002). Interconnection bandwidth and latency rank high, along with memory bandwidth, among hardware capabilities needed for effective handling of big-data applications, which are increasingly implemented using parallel and distributed processing. Considerations in this domain are discussed in the entry "Parallel Processing for Big Data."

Future Directions

The field of computer architecture has advanced for several decades along the mainstream line of analyzing general applications and making hardware faster and more efficient in handling the common case while being less concerned with rare cases which have limited impact on performance. Big data both validates and challenges this assumption. It validates it in the sense that certain data-handling primitives arise in all contexts, regardless of the nature of the data or its volume. It challenges the assumption by virtue of the von-Neumann bottleneck or memory-wall notions discussed earlier. The age of big data will speed up the process of trickling down of architectural innovations from supercomputers, which have always led the way, into servers or even personal computers, which now benefit from parallel processing and, in some cases, massive parallelism.

Several studies have been performed about the direction of computer architecture in view of new application domains and technological developments in the twenty-first century (e.g., Ceze et al. 2016; Stanford 2012). Since much of the processing schemes for big data will be provided through the cloud, directions of cloud computing and associated hardware acceleration mechanisms become relevant to our discussion here (Caulfield et al. 2016). Advanced graphics processors (e.g., Nvidia 2016) will continue to play a key role in providing the needed computational capabilities for data-intensive applications requiring heavy numerical calculations. Application-specific accelerators for machine learning (Sato et al. 2017), and, more generally, various forms of specialization, constitute another important avenue of architectural advances for the big-data universe.

Cross-References

- ▶ Energy Implications of Big Data
- ▶ Parallel Processing with Big Data
- ▶ Storage Hierarchies for Big Data
- ▶ Storage Technologies for Big Data

References

Aiken HH, Hopper GM (1946) The automatic sequence controlled calculator – I. Electr Eng 65(8–9):384–391

Bandi N, Sun C, Agrawal D, El Abbadi A (2004) Hardware acceleration in commercial databases: a case study of spatial operations. In: Proceedings of the international conference on very large data bases, Toronto, pp 1021–1032

Baraniuk R (2007) Compressive sensing. IEEE Signal Process Mag 24(4):118–121

Benini L, De Micheli G (2002) Networks on chips: a new SoC paradigm. IEEE Comput 35(1):70–78

Bossuet L, Grand M, Gaspar L, Fischer V, Gogniat G (2013) Architectures of flexible symmetric key crypto engines – a survey: from hardware coprocessor to

multi-crypto-processor system on chip. ACM Comput Surv 45(4):41
Casper J, Olukotun K (2014) Hardware acceleration of database operations. In: Proceedings of ACM/SIGDA international symposium on field-programmable gate arrays, Monterey, CA, pp 151–160
Cattell R (2011) Scalable SQL and NoSQL data stores. ACM SIGMOD Rec 39(4):12–27
Caulfield AM et al (2016) A cloud-scale acceleration architecture. In: Proceedings of 49th IEEE/ACM international symposium on microarchitecture, Orlando, FL, pp 1–13
Ceze L, Hill MD, Wenisch TE (2016) Arch2030: a vision of computer architecture research over the next 15 years, Computing Community Consortium. http://cra.org/ccc/wp-content/uploads/sites/2/2016/12/15447-CCC-ARCH-2030-report-v3-1-1.pdf
Chen JX (2016) The evolution of computing: AlphaGo. Comput Sci Eng 18(4):4–7
Chen CLP, Zhang C-Y (2014) Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf Sci 275:314–347
Cyrluk D, Rajan S, Shankar N, Srivas MK (1995) Effective theorem proving for hardware verification. In: Theorem provers in circuit design. Springer, Berlin, pp 203–222
Danowitz A, Kelley K, Mao J, Stevenson JP, Horowitz M (2012) CPU DB: recording microprocessor history. Commun ACM 55(4):55–63
Eggers SJ, Emer JS, Levy HM, Lo JL, Stamm RL, Tullsen DM (1997) Simultaneous multithreading: a platform for next-generation processors. IEEE Micro 17(5): 12–19
Gepner P, Kowalik MF (2006) Multi-core processors: new way to achieve high system performance. In: Proceedings of IEEE international symposium on parallel computing in electrical engineering, Bialystok, pp 9–13
Govindaraju NK, Lloyd B, Wang W, Lin M, Manocha D (2004) Fast computation of database operations using graphics processors. In: Proceedings of the ACM SIGMOD international conference on management of data, Paris, pp 215–226
Hall SH, Hall GW, McCall JA (2000) High-speed digital system design: a handbook of interconnect theory and design practices. Wiley, New York
Herlihy M, Moss JEB (1993) Transactional memory: architectural support for lock-free data structures. In: Proceedings of the international symposium on computer architecture, San Diego, CA, pp 289–300
Hilbert M, Lopez P (2011) The world's technological capacity to store, communicate, and compute information. Science 332:60–65
Hu H, Wen Y, Chua T-S, Li X (2014) Toward scalable systems for big data analytics: a technology tutorial. IEEE Access 2:652–687
Kuon I, Tessier R, Rose J (2008) FPGA architecture: survey and challenges. Found Trends Electron Des Autom 2(2):135–253
Le Gall D (1991) MPEG: a video compression standard for multimedia applications. Commun ACM 34(4): 46–58
Lee RB (1995) Accelerating multimedia with enhanced microprocessors. IEEE Micro 15(2):22–32
Lee J, Kim H, Yoo S, Choi K, Hofstee HP, Nam GJ, Nutter MR, Jamsek D (2017) ExtraV: boosting graph processing near storage with a coherent accelerator. Proc VLDB Endowment 10(12):1706–1717
Liu AX, Meiners CR, Torng E (2010) TCAM razor: a systematic approach towards minimizing packet classifiers in TCAMs. IEEE/ACM Trans Networking 18(2): 490–500
Markgraf JD (2007) The von Neumann Bottleneck. On-line source that is no longer accessible (will find a replacement for this reference during revisions)
McKee SA (2004) Reflections on the memory wall. In: Proceedings of the conference on computing frontiers, Ischia, pp 162–167
Mueller R, Teubner J, Alonso G (2012) Sorting networks on FPGAs. Int J Very Large Data Bases 21(1):1–23
Nvidia (2016) Nvidia Tesla P100: infinite compute power for the modern data center – technical overview. On-line document. http://images.nvidia.com/content/tesla/pdf/nvidia-teslap100-techoverview.pdf. Accessed 18 Feb 2018
Owens JD et al (2008) GPU computing. Proc IEEE 96(5):879–899
Parhami B (1999) Chapter 7: Sorting networks. In: Introduction to parallel processing: algorithms and architectures. Plenum Press, New York, pp 129–147
Parhami B (2005) Computer architecture: from microprocessors to supercomputers. Oxford University Press, New York
Pirsch P, Demassieux N, Gehrke W (1995) VLSI architectures for video compression – a survey. Proc IEEE 83(2):220–246
Rau BR, Fisher JA (1993) Instruction-level parallel processing: history, overview, and perspective. J Supercomput 7(1–2):9–50
Rixner S (2001) Stream processor architecture. Kluwer, Boston
Sato K, Young C, Patterson D (2017) An in-depth look at Google's first tensor processing unit, google cloud big data and machine learning blog, May 12. On-line document. http://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu. Accessed 18 Feb 2018
Scott ND, Olsen DM, Gannett EW (1998) An overview of the visualize FX graphic accelerator hardware. Hewlett Packard J 49:28–29
Shafer J, Rixner S, Cox AL (2010) The Hadoop distributed filesystem: balancing portability and performance. In: Proceedings of the IEEE international symposium on performance analysis of systems & software, White Plains, NY, pp 122–133
Shaw DE (1982) The non-von supercomputer, Columbia University technical report, on-line document. http://academiccommons.columbia.edu/catalog/ac:140914. Accessed 18 Feb 2018

Singer G (2013) The history of the modern graphics processor, TechSpot on-line article. http://www.techspot.com/article/650-history-of-the-gpu/. Accessed 18 Feb 2018

Sklyarov V et al (2015) Hardware accelerators for information retrieval and data mining. In: Proceedings of the IEEE conference on information and communication technology research, Bali, pp 202–205

Smith AJ (1982) Cache memories. ACM Comput Surv 14(8):473–530

Smotherman M (2010) IBM stretch (7030) – aggressive uniprocessor parallelism. On-line document. http://people.cs.clemson.edu/~mark/stretch.html. Accessed 18 Feb 2018

Stanford University (2012) 21st century computer architecture: a community white paper, on-line document. http://csl.stanford.edu/∼christos/publications/2012.21stcenturyarchitecture.whitepaper.pdf. Accessed 18 Feb 2018

Storer J (1988) Data compression. Computer Science Press, Rockville

Taylor DE (2005) Survey and taxonomy of packet classification techniques. ACM Comput Surv 37(3):238–275

von Neumann J (1945) First draft of a report on the EDVAC, University of Pennsylvania. On-line document. https://web.archive.org/web/20130314123032/http:/qss.stanford.edu/∼godfrey/vonNeumann/vnedvac.pdf. Accessed 14 Feb 2018

von Neumann J, Burks AW, Goldstine HH (1947) Preliminary discussion of the logical design of an electronic computing instrument. Institute for Advanced Study, Princeton

Ward MO, Grinstein G, Keim D (2010) Interactive data visualization: foundations, techniques, and applications. CRC Press, Natick

Wulf W, McKee S (1995) Hitting the wall: implications of the obvious. ACM Comput Archit News 23(1):20–24

Yoon C-W, Woo R, Kook J, Lee S-J, Lee K, Yoo H-J (2001) An 80/20-MHz 160-mW multimedia processor integrated with embedded DRAM, MPEG-4 accelerator, and 3-D rendering engine for mobile applications. IEEE J Solid State Circuits 36(11):1758–1767