Recent Advances In Computer Architecture: The opportunities and challenges for provenance

Nikilesh Balakrishnan Thomas Bytheway Lucian Carata
Oliver R. A. Chick James Snee Sherif Akoush Ripduman Sohan
Margo Seltzer* Andy Hopper

University of Cambridge, firstname.lastname@cl.cam.ac.uk
Harvard University*, margo@eecs.harvard.edu*

Abstract

In recent years several hardware and systems fields have made advances in technology that open new opportunities and challenges for provenance systems. In this paper we look at such technologies and discuss the implications they have for provenance. First, we discuss processor and memory controller technologies that enable fine-grained lineage capture, resulting in more precise and accurate provenance. Then, we look at programmable storage, 3D memory and co-processor technologies discussing how lineage capture in these heterogeneous environments results in richer and more complete provenance. We finally look at technological advances in the field of networking, namely NFV and SDN, discussing how these technologies enable easier provenance capture in the network.

Categories and Subject Descriptors 500 [*Information systems*]: Data provenance

General Terms provenance, lineage

Keywords provenance, lineage

1. Introduction

The vision for provenance research is to ultimately make provenance pervasive, seamless and a fundamental construct of future digital systems. A decade's worth of research in building general purpose systems has worked towards this vision by taking a purely software centric approach to capturing and processing provenance data. However, the main limitation of these systems is that they capture coarse-grained, domain specific and in many cases incomplete provenance data, thereby leading to provenance that lacks precision and completeness. These limitations arise mainly because software based approaches [20–22] trade-off capture granularity for runtime performance and capture completeness for system complexity. We believe that for future systems to be useful in the real world these limitations must be mitigated by capturing more *precise*, *accurate* and *complete* provenance. How do we implement systems that have these properties while keeping overheads low? The answer to this question may be in some of the recent advances in hardware technology.

Precision Precise provenance provides better confidence in the lineage of data and computations. This requires capturing fine-grained, high fidelity provenance. An interesting aspect to observe with provenance capture is that it naturally overlaps with other areas of computer science that perform similar functions such as debugging, logging, auditing and deterministic record/replay. While traditionally such functions have been implemented in software, recent trends in technology suggest that there is increasing support for these functions in hardware mainly because hardware implementations impose negligible runtime overheads on user applications. For example, modern Intel processors provide hardware assistance for instruction level program tracing while incurring a negligible amount of runtime overhead [30]. The traces captured with this technology can be used to construct byte level lineage, time travel to a past program state or determine linkages between data artifacts more accurately. Similarly, deep packet inspection hardware allows us to capture and inspect individual network packets at line rate, giving us the ability to track the lineage of every packet in a network. As implementers of provenance systems we can leverage the fine-grained capture capabilities these technologies provide for more precise and accurate provenance.

Completeness Completeness of provenance requires systems that capture lineage information in two dimensions: *within hosts* and *across hosts*.

Within hosts, we are observing a major change in hardware architectures as energy usage and application performance become critical aspects of computers. The traditional way of thinking about computing on a host has been CPU centric where data is fetched from storage or memory, moving it closer to the CPU on which computation is performed. However with the increased programmability offered by modern devices such as SSDs and memory interfaces we are seeing a change from the traditional form of computing. For example, a wealth of research is available in the area of Near Data Processing (NDP) wherein specific computations can be directly executed on the low frequency wimpy cores used in storage devices [6]. This trend can also be observed in domains using HPC, such as scqience and financial modelling where applications offload computations onto co-processors to exploit the high degree of parallel processing capabilities present in them. Thus for provenance to be complete and pervasive, systems should be adapted to capture provenance in heterogeneous environments present within hosts.

TaPP 2015, July 8–9, 2015, Edinburgh, Scotland.

Across hosts, provenance systems have generally lacked observability. This is mainly because network devices such as routers, load balancers, firewalls etc. have traditionally been implemented using specialised hardware, making integration of provenance systems with these devices hard. However, today, as scalability and network service management become critical, there is more emphasis on using commodity hardware that run software implementing various network functions. For example, Network Functions Virtualisation (NFV) is a recent advance in network architecture where several important networking functions can be decoupled from proprietary hardware and run in software [3]. This allows provenance systems to be easily integrated with networking software, thereby enabling network provenance capture and making provenance more complete.

In the rest of this paper we identify specific technological advances that we can leverage, explain why they are interesting for the provenance community, discuss various challenges these technologies impose and look at some exciting use cases they enable. Our aim is to stimulate debate, capture the imagination of the researcher and identify opportunities/challenges. To this end, we focus on three areas (i) Fine-grained capture, (ii) Heterogeneous computing and (iii) Network provenance, that are traditionally under-represented in the collection of provenance information.

2. Fine-grained capture

Fine-grained provenance capture provides a detailed image of an application's execution. The captured data can include details about individual instructions executed, memory read/write operations and system calls performed. In the sections below we look at two main hardware technologies, (i) Processor Trace and (ii) Programmable Memory Controllers, that enable us to capture fine-grained provenance on a host.

2.1 Processor Trace

With the end of Dennard's scaling we have seen a fundamental shift in processor architecture. To keep up with Moore's law the industry has been forced to change processor design by integrating multiple cores while keeping frequency approximately the same. To obtain improvements in application performance developers have had to exploit the inherent parallelism offered by multi-core architectures, thus increasing the complexity of applications and making debugging/performance tuning large applications a difficult problem.

For years debugging and performance optimisation of applications have been carried out using software based tracing and instrumentation techniques. But the industry has quickly realised that a purely software based approach is not scalable and tracing/instrumentation cannot be always-on since the run-time overhead imposed on applications can be quite high.

Modern processors address this problem by providing profiling and debugging support in hardware. To this end, some recent hardware assisted debugging technologies currently being adopted by the industry are Intel Processor Trace (Intel PT) and ARM Coresight, both of which enable the capture of fine-grained program traces with minimum performance impact [30, 31]. This program trace data along with hardware performance counter information stored in special-purpose registers provide the ability to build powerful debugging and performance analysis tools.

This bodes well for provenance as systems can leverage these capture technologies. Intel's Processor Trace technology is especially of interest for provenance capture and is discussed in detail below.

2.1.1 Intel PT

Processor Trace (PT) is an Intel architecture extension that captures software execution traces using dedicated hardware with minimal performance perturbation. The initial implementation of PT captures program control flow information. The trace information is collected in data packets which record details of program flow such as conditional branches taken/not taken, target instruction pointers for indirect branches, exceptions, interrupts and asynchronous events along with other contextual and timing information. Intel PT also provides control and filtering capabilities to customise the trace data collected. These can be programmed via a set of model specific registers (MSRs). The traces generated are output to operating system provided circular buffers in memory. Applications having access to the trace data can decode the data packets using a decoder library and reconstruct the exact execution of programs.

Given the usefulness of such debugging functionality a more recent technology offered by Intel is Trace Hub, which is intended as a complete system debug solution in hardware [29]. Trace Hub hardware is a set of functional blocks connected via PCI that can receive trace data from multiple sources such as internal hardware signals, performance counters, software traces such as PT and application/OS generated output. The data from these sources are ordered, timestamped, encoded and sent to a user configured destination such as memory, USB or MIPI PTI port.

Applicability to provenance With instruction level traces we have the lowest level view of what happened during program execution. This allows us to time travel back to a previous program state and look at the runtime context around that state. This capability allows us to reason about program behaviour, determine causes for program crashes or debug stack corruptions, thus enabling a variety of uses for provenance in debugging and troubleshooting.

With merely coarse-grained (system call level) provenance capture it is not possible to accurately determine dependencies between data inputs and outputs. However, when system call information is combined with PT trace data and program inputs, we have accurate and precise information required to retrospectively determine data flow within the program, thereby allowing us to eliminate false positive dependencies between data. The added advantage with this approach is that data flow analysis can be decoupled from the actual program execution and taken off-host for processing or done lazily during query time.

Challenges Precision and completeness are traded off with performance when implementing Intel PT tracing. The mechanism can generate vast amounts of data quickly, approximately 200 MB/s per core running at 1600 MHz. However, it comes at the cost of impacting the the compute path as it results in processor slowdown. Furthermore, the storage footprint of PT traces may directly affect storage capacity and limit bandwidth especially when full PT data capture is enabled for multiple machines in a cluster and taken off host for processing.

The data generated by Intel PT is already compressed, for example, conditional branches generate a single bit to indicate taken or not taken. Therefore standard compression techniques may not be effective. This opens up some interesting research opportunities in storing, processing and querying such data for future provenance systems that use this technology.

Finally, the data gathered using Intel PT is low level (instruction level) and does not contain the high level semantics that normal users would care about. Therefore, for many users this low level view of their application may not be useful. An interesting research problem is to try and map high level semantic concepts to these low level traces.

2.2 Programmable Memory Controllers

Provenance systems that use system/library call interception mechanism to collect provenance lack the ability to observe memory operations performed by applications since memory operations typically bypass the OS system call interface. However, all memory read and write operations are handled and managed by the memory controller. The memory controller runs firmware that implements several sophisticated functions, with read/write management being one of the functions. The firmware running on most commercial memory controllers is generally not programmable as the firmware is proprietary. However, previous research shows that allowing memory controller's firmware to be programmed based on system and application requirements improves its versatility and efficiency [15]. This ability can be exploited by provenance systems to run custom memory read/write capture functionality as part of the controller firmware, giving systems observability into application's memory operations.

Applicability to provenance Programmable memory controllers can be used as a viable side channel for provenance collection. The advantage of using such side channels for provenance capture is (i) There is zero overhead on the application developer and, (ii) The entire system (applications and OS) can be completely agnostic about provenance collection.

In addition to the advantages mentioned above, this technology enables several use cases. The provenance collected can include individual load/store operations performed by applications and/or snapshots of application memory state. This data can be used to reason about system behaviour, for example, recent work shows that memory heat maps can be used to learn about normal system behaviour and deviations in behaviour can be detected by comparing memory heat maps obtained from subsequent executions [23]. Load/store patterns or memory heat maps can also be used for performance optimisations (e.g. accessing contiguous memory regions is faster). Memory state/deltas captured can allow debuggers to time travel and replay execution from arbitrary points in the code.

Challenges As with tracing, enhanced completeness and precision come at a large storage footprint cost. The data collected by memory controllers is at a low level of abstraction and the size of data captured can grow rapidly, thus affecting storage capacity. Provenance systems cannot persist this data forever.

Secondly, affecting the performance of the memory controller negatively may impact the entire system since it is in the critical path of all programs executing on the machine. Completeness can be compromised if only a subset of accesses are recorded.

Finally, the load/store addresses captured by the memory controller are physical addresses, therefore we may need additional information from the OS to determine virtual to physical address mappings and associate load/store operations to individual applications. Without additional system-level information mapping accesses to identifiable abstractions completeness will be compromised.

3. Heterogeneous Computing

Modern hardware architecture has become increasingly heterogeneous, comprising of multiple processing elements (typically running different ISAs), each specialised to perform specific functions. This shift to heterogeneous architectures mainly began with the introduction of user programmable GPUs about a decade ago as developers sought improvements in performance of their applications (especially in HPC). Today, a variety of GPU type devices or co-processors are commonly used in various domains.

However, heterogeneous system architecture goes beyond just co-processors. Currently, there is a resurgence in Near Data Processing (NDP) research [6]. NDP changes the computational paradigm by moving computation closer to where data resides. This provides benefits in terms of reduced power consumption and increased performance as data does not have to move back and forth between storage and CPU. Today, we are seeing the emergence of devices such as programmable storage/memory, which allow custom code to be executed directly on the device, closer to data [14, 16].

In the sections below we will look at three specific technologies in heterogeneous computing, (i) Programmable SSDs, (ii) 3D Packaged Memory and (iii) Co-Processors, along with the implications they have for provenance completeness.

3.1 Programmable SSD

Modern SSDs contain several low frequency CPUs running firmware that perform low level functions such as error-correction, wear levelling, read/write caching and encryption to name a few. However, most SSD firmware is proprietary and does not allow developers to re-program or add new functionality. Recent work exposes new interfaces that developers can use to program SSDs [8, 9, 14]. These interfaces allow application developers to offload certain application specific functionality to run directly on the SSD. This offers huge improvements in bandwidth, access latencies and reduces power consumption as data does not have to move back and forth between the SSD and the host. This basic primitive also allows higher level functions such as implementing high performance data stores, data intensive computing on the SSD, kernel bypass for user space applications and semantic extensions.

Applicability to provenance Provenance-aware systems typically run on the host CPU either as part of the operating system or the application runtime. When data transformations are performed by applications directly on the SSD, the transformations will not be visible to the provenance collection system running on the host CPU, making provenance capture incomplete. Collecting and integrating provenance in these heterogeneous environments makes provenance capture more complete.

Challenges The compute units within SSDs have limited processing power. This is likely to remain the same in future due to low power consumption requirements. Adding provenance capture functionality to run on the SSD can incur overheads on I/O performance, negating the performance benefits provided by near data processing. The other challenge is in integrating and storing the captured provenance data. Provenance capture and processing components should somehow link the provenance captured on the host (CPU) with that captured on the SSD. Furthermore, block-level capture may not be granular enough to increase overall provenance precision. Tackling this issue requires identifying (and persisting) changed byte-ranges in blocks. This is a spatially and temporally expensive exercise.

3.2 3D Packaged Memory

Over the last couple of decades we have seen a growing disparity between processor and off-chip memory speeds. In the industry this is referred to as the "Memory Wall" problem where access to the memory subsystem has been drastically affected in terms of both bandwidth and latency [26]. The problem has been further aggravated due to multi/many-core processors as they have imposed increased demands on the processor/memory interface. This has led to research in 3D packaged memory where two or more RAM chips are stacked one on top of each other and connected using Through-Silicon Vias (TSV) [24, 25]. 3D memory technology increases the memory capacity and alleviates the memory bandwidth and latency

problem. In addition, 3D packaging also offers the possibility of integrating compute logic directly with one or more memory layers. This once again changes the computing paradigm as it opens up the possibility of near-memory computations. This idea is not new and there is a body of research available on Processing In Memory (PIM) [17, 18]. As research in NDP and PIM evolves it is inevitable that we will have fully-programmable memory architectures, where application developers can run custom code directly near memory.

Applicability to provenance Similar to programmable SSDs, with 3D packaged memory the provenance collection system running on the CPU will lack visibility into the computations and data transformation that have taken place on the compute logic close to memory, thereby making provenance collection incomplete. Collecting provenance in this computational environment improves the completeness of provenance.

Challenges The Processing In Memory (PIM) compute paradigm can involve having fixed-function compute logic directly integrated with memory or expose the compute logic as a fully-programmable layer to developers [19]. Fixed-function PIMs may not provide the flexibility to run provenance capture on the compute layers near memory as it will require implementers to disclose provenance as part of code being executed. This poses provenance capture challenges for systems. However, there will be more flexibility in case of fully-programmable PIMs as they provide similar programmability as conventional processors. While PIMs can increase the completeness and precision of provenance collection by allowing applications to offload provenance-sensitive computations to PIMs recording provenance information it is essential captured provenance can be linked back to the original application to prevent semantic identification issues as highlighted previously.

3.3 Co-Processors

There are several types of co-processor technologies prevalent today in computing. Two of the most commonly used technologies are General-purpose Graphic Processing Units (GPGPU) and Many Integrated Cores (MIC). Co-processors offer a high degree of parallel processing capabilities as they use several hundreds of simple architecture, low frequency cores. These devices are typically connected to the CPU through a high speed interface such as PCIe. To utilise the co-processors, developers have to explicitly copy data from CPU memory to co-processor memory and invoke computational kernels to process the data. After performing the computations, the results are copied back to the CPU.

This computational paradigm is commonly used in the HPC applications as the workloads are highly parallel and compute heavy. However, we are seeing an increased use of co-processors today in big data and machine learning type workloads [27]. This trend is expected to continue as applications from other domains find the massive parallel processing capabilities present in co-processors useful.

Applicability to provenance Most co-processors have their own memory. The computations and data transformations happening in co-processor memory is typically asynchronous and not visible to the operating system or the provenance collection system on the host CPU. Also co-processors such as Intel Xeon Phi's can mount and use block devices attached to the external host system [28]. Any data updates made by the co-processor directly on the mounted file system will not be visible to provenance systems running on the host CPU. Therefore co-processors should be augmented to run services that collect provenance. Collecting data lineage in this computational framework makes provenance richer and more complete.

Challenges Co-processors are used mainly to improve the performance of applications. Capturing provenance in the critical path especially at finer granularity can directly impact the performance advantages these technologies provide.

4. Network Provenance

In recent years there has been a developing trend in the networking community of systems which reduce the complexity of single entities in a network in favour of several smaller entities. Two results of this trend are the technologies, Software Defined Networking (SDN) and Network Functions Virtualisation (NFV) [3–5].

4.1 Software Defined Networking

Software Defined Networking (SDN) is an approach where the control and data planes of the network are separated out entirely, reducing the switching devices to dumb end points and moving the routing and other control activities into dedicated hosts. This separation of control and the availability of programmable systems such as OpenFlow switches has resulted in recent work where SDN is leveraged to capture provenance of individual packets for network diagnosis [1].

4.2 Network Functions Virtualisation

NFV applies this same reduction of complexity approach to the variety of special purpose devices such as firewalls, intrusion detection devices, NAT translators which are implemented on custom hardware running proprietary software. However as scalability and management of these devices become critical, NFV endeavours to implement these functions on virtual servers running on enterprise hardware, thereby eliminating the tight coupling these functions had with vendor proprietary hardware and software.

Applicability to provenance The shift towards commodity hardware implementing important network functions provides the opportunity to run provenance systems as a service on these devices. This also reduces the design and development effort as several physically distinct devices can now be implemented as virtual devices running on the same end-host server. Therefore, there is an opportunity to collect richer, more complete provenance with lower overhead and gain visiblity across hosts.

Challenges The major challenges associated with capturing provenance in the context of NFV and SDN are centred around the fact that these are still emerging technologies. Therefore at the moment there is only moderate adoption outside of academic contexts along with several inherent technological challenges to overcome [2]. In particular, ensuring completeness across multiple hosts requires synchronisation and co-ordination across all hosts involved in a network transfer. Furthermore, precision is limited by the observable information disclosed or observed at the end-host.

5. Use Cases

We outline three example use-cases highlighting the usefulness of leveraging hardware for increased provenance precision and completeness:

Real-time, always-on isolation of downloads: Employing hardware leveraged provenance across an enterprise network allows administrators to maintain a system-wide demarcation between trusted and untrusted data. Provenance information can be utilised to ensure that downloaded data is always accessed and modified in an isolated environment (e.g. a sandbox) to prevent payload malware from accessing or modifying external system state.

Feed-forward Computation: Collecting precise and complete provenance renders feasible computation platforms where improvements in data or computation can be fed *forward* to update

existing outputs thereby ensuring that outputs always reflect transformations of the most up-to-date input data.

Enabling Statistical Equivalence: Reproducibility in current contexts is limited to the deterministic reproduction of an existing computation. With precise and complete provenance information we enable the creation of equivalence engines that can rely not only on deterministic inputs and outputs and program control flow but on the data provenance and control flow provenance collected during the creation of those outputs. This allows practitioners to broaden the definition of reproducible equivalence from the narrow aspect of equivalent input-output pairs.

6. Discussion

Over this paper we have seen how technology advances provide both opportunities and challenges for provenance. Hardware technologies that enable fine-grained capture result in provenance that is more precise and accurate but result in the collection of vast amounts of data that is challenging to process and store.

Heterogeneous architectures provide new opportunities for provenance collection resulting in richer more complete capture but at the moment there is little effort to build systems that capture provenance in these computational frameworks.

Network architecture and management of networks are becoming more software centric as we have seen with NFV and SDN technologies. This augur's well for provenance systems as these technologies make it easier to integrate and capture provenance in the network, where up until now we have had limited observability. However these networking technologies are currently at a fledgling state, thereby creating integration challenges for provenance systems.

Our vision for provenance is that it ultimately becomes ubiquitous, transparent and easily accessible for users or applications. We have shown that to realise this vision, we as implementers of systems should be cognizant of advances in technology, leverage the opportunities provided by these technologies and address the challenges technologies pose.

7. Conclusion

In this paper we discussed several technological advances that create both opportunities and challenges for future provenance systems. We talked about a few exciting technologies that enable fine-grained provenance collection and how leveraging these technologies makes provenance more precise and accurate. We also discussed areas that are currently under-represented in provenance and how lineage capture in these areas makes provenance more richer and complete.

References

[1] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazires, Nick McKeown, "I know what your packet did last hop: using packet histories to troubleshoot networks", Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14). USENIX Association, Berkeley, CA, USA, 71-85.

[2] Han, Bo, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. "Network function virtualization: Challenges and opportunities for innovations." Communications Magazine, IEEE 53, no. 2 (2015): 90-97.

[3] M. Chiosi et al. Network Functions Virtualisation: An Introduction, Benets, Enablers, Challenges & Call for Action. ETSI White Paper, Oct. 2012.

[4] The European Telecommunications Standards Institute. Network Functions Virtualisation (NFV); Architectural Framework. GS NFV 002 (V1.1.1), Oct. 2013.

[5] The European Telecommunications Standards Institute. Network Functions Virtualisation (NFV); Use Cases. GS NFV 001 (V1.1.1), Oct. 2013.

[6] Rajeev Balasubramonian, Jichuan Chang, Troy Manning, Jaime H. Moreno, Richard Murphy, Ravi Nair, Steven Swanson, "Near-Data Processing: Insights from a MICRO-46 Workshop", IEEE Micro, vol.34, no. 4, pp. 36-42, July-Aug. 2014, doi:10.1109/MM.2014.55

[7] Diego Kreutz, Fernando M. V. Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, Steve Uhlig, "Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE, vol.103, no.1, pp. 14-76, Jan. 2015, doi:10.1109/JPROC.2014.2371999

[8] Sudharsan Seshadri, Mark Gahagan, Sundaram Bhaskaran, Trevor Bunker, Arup De, Yanqin Jin, Yang Liu, Steven Swanson, "Willow: A User-Programmable SSD", Proceedings of OSDI. 2014, 67-80.

[9] Arup De, Maya Gokhale, Rajesh Gupta, Steven Swanson, Minerva: Accelerating Data Analysis in Next-Generation SSDs, Proceedings of the 21st IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), April 2013.

[10] Matthew Breitwisch, Phase change memory, Interconnect Technology Conference, 2008. IITC 2008. International, pages 219221, June 2008

[11] Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer, "Provenance-aware storage systems", USENIX Annual Technical Conference, 2006.

[12] Nikilesh Balakrishnan, Thomas Bytheway, Ripduman Sohan, Andy Hopper, "OPUS: A Lightweight System for Observational Provenance in User Space", Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, Berkeley, CA, USA, series TaPP '13, pp. 8:1–8:4, USENIX Association, Apr 2013.

[13] Ashish Gehani, Dawood Tariq, "SPADE: support for provenance auditing in distributed environments", Proceedings of the 13th International Middleware Conference, 2013, 101120.

[14] Jaeyoung Do, Yang-Suk Kee, Jignesh M. Patel, Chanik Park, Kwanghyun Park, and David J. DeWitt, "Query processing on smart SSDs: opportunities and challenges", In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD '13). ACM, New York, NY, USA, 1221-1230. DOI=10.1145/2463676.2465295 http://doi.acm.org/10.1145/2463676.2465295

[15] Mahdi Nazm Bojnordi, Engin Ipek, "A programmable memory controller for the DDRx interfacing standards", ACM Trans. Comput. Syst. 31, 4, Article 11 (December 2013), 31 pages, 2013. DOI=10.1145/2534845 http://doi.acm.org/10.1145/2534845

[16] Qi Guo, Nikolaos Alachiotis, Berkin Akin, Fazle Sadi, Guanglin Xu, Tze Meng Low, Larry Pileggi, James C. Hoe, Franz Franchetti, "3D-Stacked Memory-Side Acceleration: Accelerator and System Design", 2nd Workshop on Near Data Processing (WONDP) in conjunction with the 47th IEEE/ACM International Symposium on Microarchitecture (MICRO-47) Cambridge, UK, 2014

[17] David Patterson, Thomas Anderson, Neal Cardwell, Richard Fromm, Kimberly Keeton, Christoforos Kozyrakis, Randi Thomas, Katherine Yelick, A Case for Intelligent DRAM: IRAM, IEEE Micro, vol. 17(2), April 1997

[18] Mary Hall, Peter Kogge, Jeff Koller, Pedro Diniz, Jacqueline Chame, Jeff Draper, Jeff LaCoss, John Granacki, Jay Brockman, Apoorv Srivastava, William Athas, Vincent Freeh, Jaewook Shin, Joonseok Park, Mapping Irregular Applications to DIVA, a PIM-based DataIntensive Architecture, Proceedings of SC, 1999

[19] Gabriel H. Loh, Nuwan Jayasena, Mark H. Oskin, Mark Nutter, David Roberts, Mitesh Meswani, Dong Ping Zhang, Mike Ignatowski, "A Processing in Memory Taxonomy and a Case for Studying Fixed-Function PIM", WoNDP: 1st Workshop on Near-Data Processing. (2013)

[20] Peter Macko and Margo Seltzer, "A General-Purpose Provenance Library", Theory and Practice of Provenance, 2012

[21] David Allen, Adriane Chapman, Barbara Blaustein, Len Seligman, "Provenance Capture in the Wild", IPAW, 2010

[22] Blake Coe, Christopher Doty, David Allen, Adriane Chapman, "Provenance Capture Disparities Highlighted through Datasets", Theory and Practice of Provenance, 2014.

[23] Man-Ki Yoon, Sibin Mohan, Jaesik Choi, Lui Sha, Memory Heat Map: Anomaly Detection in Real-Time Embedded Systems Using Memory Behavior, to appear in the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC 2015), Jun. 2015.

[24] Joonyoung Kim and Younsu Kim, "HBM: Memory Solution for Bandwidth-Hungry Processors", 2014, HC26, Hot Chips: A Symposium on High Performance Chips.

[25] Todd Farrell, HMC Overview: A Revolutionary Approach to System Memory, 2012, exhibit at Supercomputing.

[26] Wm A. Wulf, Sally A. McKee, "Hitting the memory wall: Implications of the obvious", Computer Architecture News 23 (1) (1995) 2024.

[27] Big Data Analytics, Data Science & Machine Learning - Nvidia, http://www.nvidia.com/object/data-science-analytics-database.html

[28] Assigning physical hard drives to Intel Xeon Phi coprocessors, https://software.intel.com/en-us/articles/assigning-physical-hard-drives-to-intel-xeon-phi-coprocessors

[29] Intel Trace Hub, Developer's Manual, March 2015, Revision 1.0, https://software.intel.com/sites/default/files/managed/d3/3c/intel-th-developer-manual.pdf

[30] Intel 64 and IA-32 Architectures Software Developers Manual, Chapter 36, vol.3C, pages 36.1-36.37, http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf

[31] CoreSight Debug and Trace - ARM, http://www.arm.com/products/system-ip/debug-trace/