Netherlands Organisation
for Scientific Research

Vernieuwingsimpuls Vernieuwingsimpuls / Innovational Research Incentives Scheme
Grant application full proposal form 2019
Social Sciences and Humanities, Applied and Engineering Sciences
Please *use the Explanatory Notes when completing this form* **Veni scheme**

Research proposal

2a1, 2a2, and 2a3. Description of the proposed research

2a1. Overall aim and key objectives

Computers power our Digital Economy. We need, and take for granted, that software and hardware advancements will keep making computers faster and more scalable. This assumption held true for many decades, as we relied on Moore's Law to deliver faster computers by building a platform of "*powerful CPUs and slow devices*". However, as Moore's Law has slowed down (thus stalling the CPU performance improvements), our **CPU-centric** software frameworks are struggling to deliver the expected performance [1-2]. Consequently, we have begun to innovate rapidly toward a diverse set of fast, non-CPU, compute and Input-Output (I/O) *accelerator devices* [3], e.g. Tensor Processing Units [4], Graphics Processing Units (GPUs) [5], smart networks [6-8], storage [9-10], FPGAs [11], etc. These devices deliver performance through *semi-specialized*, *programmable* hardware-circuits rather than through general-purpose, CPU-centric processing. We experience now a *Cambrian innovation-explosion* of heterogeneous platforms with "*weaker CPUs and fast devices*" [12].

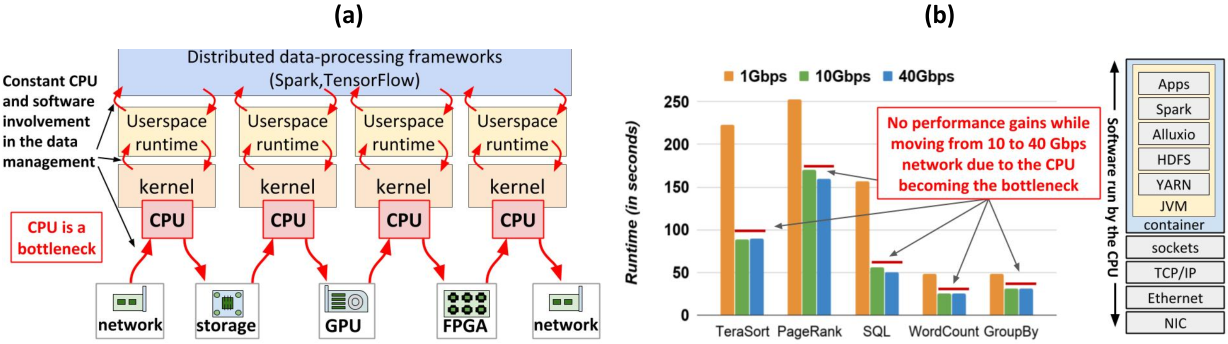


Figure-1: (a) The current CPU-centric architecture; **(b)** its effects on the runtimes (vertical axis, lower is better) of data-processing workloads (horizontal axis), even with high-performance networks [13].

However, despite significant innovations in hardware, our software data-processing frameworks and their application programming interfaces (APIs) have not evolved as rapidly, partially due to the convenience of the CPU-centric architecture. In this architecture (Figure-1(a)), the CPU, systems software (kernel, runtime) and data-processing frameworks are constantly involved in managing devices, while simultaneously coordinating *dataflows* between them. In my research [13-15], I have demonstrated that, as devices get faster, this architecture leads to significant performance losses (Figure-1(b)). The key research challenge is *how to leverage modern, heterogeneous, accelerator devices to enable the fast and efficient computing needed by our data-driven Digital Economy.* So far, the community has tried:

- **New hardware, old CPU-centric software:** To improve the CPU-centric architecture, the community tried to optimize only the API implementation for new hardware [16-17]. Though this delivers (marginal) performance gains [18-19], it still requires CPU-centric coordination between devices - a significant bottleneck. As I have demonstrated, the CPU is 100% occupied [13,20] even as accelerator-devices remain under-utilized, greatly limiting performance gains (Figure-1(b)).

- **New hardware, new software:** Many new *built-from-scratch software* for distributed storage [21-23] (including mine [24-26]), transactions [27-28], communication [28], and operating-systems [30] has been designed to leverage new hardware. Though going in the right direction, these approaches (i) are not general-purpose, focus on particular workloads or restricted setups, e.g. "transactions" on a battery-backed DRAM or single-machine device-CPU optimizations [31-32]; and (ii) built-from-scratch software requires significant efforts to develop toolchains and ecosystems, and to create critical mass among developers from scratch.

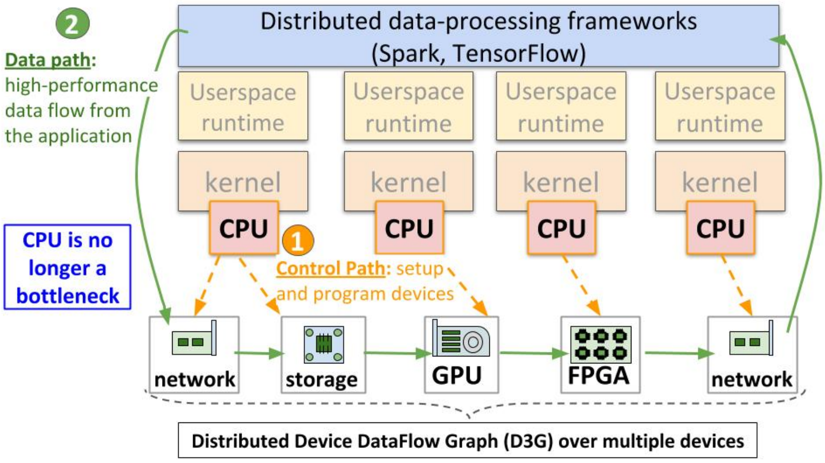

2
Data path: high-performance data flow from the application
Distributed data-processing frameworks (Spark, TensorFlow)
Userspace runtime
Userspace runtime
Userspace runtime
Userspace runtime
kernel
kernel
kernel
kernel
CPU
CPU
CPU
CPU
CPU is no longer a bottleneck
1
Control Path: setup and program devices
network
storage
GPU
FPGA
network
Distributed Device DataFlow Graph (D3G) over multiple devices

Figure-2: Versus Figure-1(a), Hermes's "device-centric architecture" splits control and data paths.

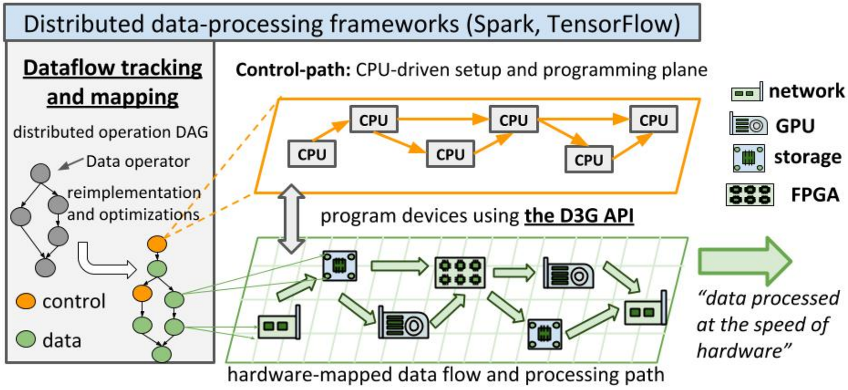
The overall aim of **Project Hermes** is to leverage heterogenous accelerator-devices, by
(goal **G1**) developing a general-purpose, "*fast-by-design*" framework, which
(**G2**) *systematically* removes the CPU from data-processing, while
(**G3**) maintaining much of the established development ecosystem and
(**G4**) in particular, remaining cloud-ready.

Hermes proposes a "***device-centric computing architecture***", which, **by-design**, puts modern hardware-devices instead of the CPU at the center of data-processing. This device-centric design delivers performance by offloading data-management responsibilities, network transfers, storage accesses, "specialized" data-processing routines, to modern accelerator-devices. My key insight is that, for the common abstraction of *dataflow* [33-34], which is used in many popular data-processing frameworks (e.g., Spark, TensorFlow, and Flink), we can use the principle of splitting data and control paths, to shift (offload) work from the CPU to modern programmable devices (Figure-2). The CPU is not eliminated, but only runs the control path responsible for allocating resources and programming devices, and ideally only at processing-start. Consequently, *most* data-processing happens in non-CPU hardware, by leveraging accelerator-devices. This new design raises new challenges:
(challenge **C1**) How to programmatically coordinate dataflows on heterogeneous, distributed devices?
(**C2**) How to control and manage heterogeneous devices in a dataflow framework?
(**C3**) How to deliver performance while managing complexity from heterogeneous devices in the cloud?

To address these challenges, my research adheres to **accepted methodologies in computer systems research**:
(methodology **M1**) Quantitative research (modeling, simulation, surveys) [35-36];
(**M2**) Design, abstraction, prototyping [37-38];
(**M3**) Experimental research, designing appropriate benchmarks [39-40];
(**M4**) Use-case study, collecting operational traces (following best practices [41-44]);

(**M5**) Open-science, open-source software, community building, peer-reviewed scientific publications [45].

Figure-3: The Herr


Distributed data-processing frameworks (Spark, TensorFlow)
Dataflow tracking and mapping
distributed operation DAG
Data operator
reimplementation and optimizations
control
data
Control-path: CPU-driven setup and programming plane
CPU
CPU
CPU
CPU
CPU
CPU
program devices using the D3G API
network
GPU
storage
FPGA
"data processed at the speed of hardware"
hardware-mapped data flow and processing path

Challenge-1: Programmatically coordinate dataflows on heterogeneous, distributed devices.

Modern data-processing frameworks run in a distributed setting with hundreds of servers containing many accelerator devices. The key challenge here is to build a global abstraction, and enforcing mechanisms to identify and program devices for coordinating dataflows through them in the distributed setting.

Approach: I propose to unify the treatment of multiple devices under a *new* abstraction of "*distributed dataflow device-graphs (D3Gs)*", which models how data *should* flow through devices. The key insight here is that a D3G can often be *pre-built* by the CPU on the control path to identify, coordinate, and then program devices. Then data will only flow through devices on the D3G edges at the "*speed of hardware*". I will design the D3G API to identify and transfer data over multiple devices (using **M2**), and implement and evaluate an open-source prototype (**M3, M5**). As main result, data will move through multiple distributed, heterogeneous devices without involving the CPU after processing-start (achieves **G2**).

Challenge-2: Manage and control heterogeneous devices in a dataflow framework.

Instead of programming devices directly, scientists use dataflow-based frameworks for data processing, e.g. Spark, TensorFlow, and Flink. The key challenge remains to track dataflows *inside* these distributed frameworks, in particular tracking dynamically *which* device (should) receive *what* part of the data for processing and *when*.

Approach: I observe that such frameworks often represent workloads as a direct acyclic graph (DAG) of data-processing operators, which implicitly tracks dataflows (each operator has a well-defined input and output set), but only to provide fault-tolerance. I propose to expand this mechanism to also manage and control performance by dynamically tracking the dataflow between operators, then mapping the dataflow to distributed devices using the D3G API (**M1-2**). I will benchmark the prototype (**M3**) and collect device-utilization traces (**M4-5**, used in Challenge-3). As main result, Hermes will deliver a *general-purpose*, *device-accelerated dataflow framework* (**G1**) in a mature ecosystem, e.g. Spark (**G3**).

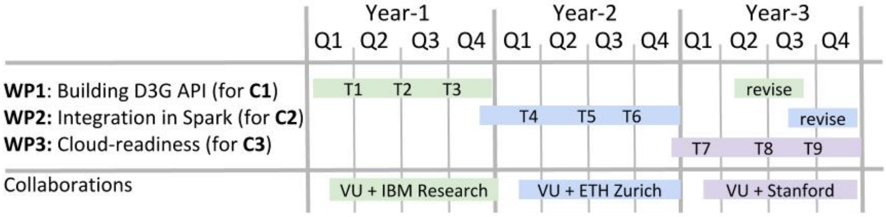
Challenge-3: Managing complexity and performance in the cloud.

Data-scientists use cloud-based data-processing frameworks for on-demand resource scaling. But acquiring devices and configuring frameworks for their best-possible performance remains a challenging task [46-47]. Recently, *serverless* computing has emerged as a way to shield users from configuration complexity by programming with small stateless functions managed by the cloud operator [48-49]. I propose to use serverless computing for dataflow data-processing. However, this entails challenges associated with performance (and cost): acquiring the *right* cloud-devices while *managing* the dataflow on behalf of users (a challenge also because functions are now assumed stateless, but dataflows require device-state and data management).

Approach: My insight is that, using the Hermes architecture, I can apply machine-learning techniques (using traces collected from **C2**) to predict the load on the control path, and predictively pre-acquire and pre-program the right devices (**M1**). I will design mechanisms (**M2**) to *store* dataflow-state in distributed, shared data-stores (e.g. Pocket, which I have previously developed in collaboration with Stanford [50-51]). The main result will be an open, elastic distributed data-store for high-performance data processing (**M3, M5**) with cloud-based serverless deployment (**G4**).

2a2. Research plan

Figure-4: Research timeline.



My research-challenges (**C1-3**) map naturally to 3 work packages (**WPs**):

WP1/C1: The D3G abstraction and API to programmatically coordinate dataflows on heterogeneous, distributed devices. (collaborators: VU and Bernard Metzler, IBM Research)

Leveraging my previous work [52-56], I will design a distributed 128-bits shared address-space to identify devices (task **T1** in Figure-4). With this addressing, I will build a D3G API, using D3G graphs with vertices representing devices and edges representing data transfers between devices (**T2** in Figure-4 and the bottom green plane in Figure-3). Data transfers will use specialized, CPU-free technologies in devices, e.g., RDMA and Peer-to-Peer DMA [57-58] (**T3**), thus accelerating dataflows to the speed of hardware (achieving **G3**).

Risks: If a desired feature is missing, I will implement that in programmable BlueField network-accelerators [7] secured specifically for project Hermes from Mellanox, a leading manufacturer of network equipments.

WP2/C2: Manage and control heterogeneous devices for a Hermes-accelerated data-processing framework, with Spark integration as demonstrator. (collaborators: VU and Thomas Gross, ETH Zurich)

I will not build a Hermes-accelerated data-processing framework from scratch, but will integrate it in Spark[1] [59-60], which will further serve as demonstrator. Building on my previous experience with Spark [56], I will

[1] Apache Spark is one of the most popular and used frameworks in the world [61-62].

synthesize dataflows by using logging and tracing in this real-world framework, e.g. for the Join or TreeReduce operators (**T4**). I will *map* the synthesized dataflows *operator-by-operator* to heterogeneous devices by re-implementing operators (e.g. sort, filter, multiply) to match device-capabilities and by pre-programming devices using the D3G API (**T5**). To optimize across the entire dataflow, I will then integrate these device-mapped operators in Spark's workload DAG for compilation in the managed runtime (**T6**) [63]. This approach builds a *Hermes-accelerated*, *general-purpose, data-processing framework* on Spark (achieving **G1-2**); similar approaches could achieve the same for TensorFlow, Flink, etc.

Risks: If an operator cannot be cleanly mapped to a device, I will leverage the alternate architecture of monotasking [64], where each data-processing tasks is mapped to one resource by design. Spark Flare architecture [65] can be utilized to reduce runtime overheads.

WP3/C3: Hermes-accelerated Spark in Serverless to manage device complexity in the cloud.
(<u>collaborators</u>: VU and Ana Klimovic, Stanford)

I will use device-utilization data from WP2 with machine-learning techniques (linear regression and decision trees [66-67]) to identify *which* devices should be acquired *on-demand* for upcoming operator-executions (**T7**). To prepare devices for the execution of (stateless) serverless functions, I will track state using a light indirection layer [68] (**T8**), and then save and restore states in a data-store like Pocket [50], using my previous work on storage formats [14] (**T9**). As a result, the Hermes-accelerated Spark can be deployed in the *cloud* without requiring users to manage devices, or dataflow-/device-states (achieving **G4**).

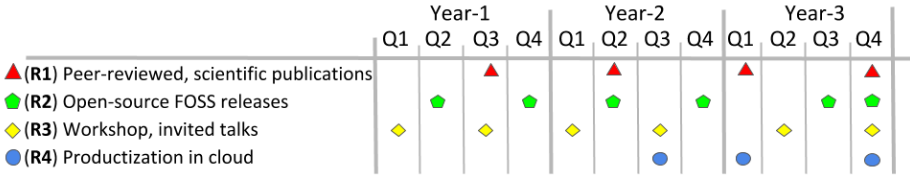
Risks: In case accelerator-devices are not available in serverless-cloud offerings, I will deploy on-premise serverless software (OpenWhisk [69]) with state-of-the-art devices to evaluate the system.

Looking beyond: As Patterson and Hennessy explain in their 2018 Turing-Award lecture [3], we are living in a golden age of hardware-software co-design. Thus, my work in Hermes paves the way for a long-term research agenda of developing *"efficient-by-design"* frameworks, by reasoning about how to leverage the best of both, the powerful hardware in accelerator-devices and the software on the CPU. The current focus is on performance, but how to include energy, security, and cost is promising future-research. Looking beyond, the emergence of neuromorphic, bio-inspired chips [70] with millions of networked neurons make a design like Hermes with its explicit dataflows and distributed state management even more appealing.

2a3. Motivation for choice of host institute

I will embed in Prof. Iosup's MCS research group and in the CompSys section at VU. The MCS group has a unique vision aligned with Hermes [71], and a strong, world-class track record (demonstrated by numerous publications and awards) in designing, deploying, and benchmarking massive distributed systems. I will be part of the international standardization organization SPEC Research's Cloud Group and get access to their workloads and traces. At VU, I will collaborate with Prof. Henri Bal on distributed programming languages and Prof. Herbert Bos on security; both are outstanding researchers. Through VU, I will have access to the NWO-funded DAS-5/-6 multi-clusters [72] for Hermes's experiments. Through MCS, I will have a leadership role in establishing and growing the research-oriented Honors Track of the VU BSc while teaching and mentoring high-quality students.

2b. Knowledge utilisation (Max. 750 words on max. two pages)



	Year-1				Year-2				Year-3			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
▲ (**R1**) Peer-reviewed, scientific publications			▲			▲			▲			▲
⬟ (**R2**) Open-source FOSS releases		⬟		⬟		⬟		⬟			⬟	⬟
◇ (**R3**) Workshop, invited talks	◇		◇		◇		◇			◇		◇
● (**R4**) Productization in cloud							●		●			●

Figure-5: Knowledge utilization timeline, showing expected results (**R1-4**) over the project duration.

Societal impact: I expect the Hermes hardware-accelerated framework will benefit the life-sciences (bioinformatics, *faster* cancer screening, *precise* personalized medicine) and fundamental sciences (Dutch participation in the SKA experiment, *accurate* climate modeling), both of which have currently *data-intensive* and *data-driven* disciplines.

Potential beneficiaries:

(a) **End-users (data analysts, scientists, students)** obtain performance today by continuously buying new hardware. This is costly and inefficient, yielding only marginally better performance (Moore's Law's effects have diminished). To get more, they currently need to understand the nuances of performance modeling and optimization for their infrastructure. Hermes aims to keep the familiar top-level data-processing tooling, and deliver performance in the cloud with software and engineering innovations through novel design, re-writing internal operators, and efficiently utilizing the current hardware (**WP2**).

(b) **Cloud providers** invest heavily to provide services and tools with high efficiency, as margins in efficiency influence profits. Hermes **WP3** aims to run data-processing services closer to hardware, thus improving both performance and utilization, and consequently the profit margins.

(c) **Computer Scientists (systems, languages, and performance researchers and engineers)** can use the principle of *distributed* data and control path-splitting to build applications and frameworks beyond data-processing. This is an active field of research, specially for emerging hardware and systems support for machine learning workloads [32,73]. The broader appeal of the Hermes architecture (**WP1** and **WP2**) is that it can be dynamically positioned to find a new balance between hardware (fast, data path) and software (slow, control path), to match specific workload requirements.

Implementation: I will ensure that the results of my research are available for potential beneficiaries through multiple channels and activities (results **R1-R4**, timeline in Figure-5). I expect knowledge utilization to start after 6 months from project-start and to continue throughout the work.

(a) **Open-sourcing and community building:** My work in Hermes will result in free open-source software (FOSS) (**R2**). Throughout the project I will update the code, fix bugs, and provide new features that will allow users to build emerging applications. Building upon my experience with the FOSS community [26,55-56], I will involve potential developers and users over multiple channels (mailing lists, slack, blogs), building a thriving community around the software artifacts, as I did for example for Crail (**M5**). I will organize hands-on workshops for users, and give talks at practitioners' venues like DataWorks/Spark summits (**R3**).

(b) **Collaboration with Dutch and international companies:** I will reach out to cloud providers (AWS, Microsoft, IBM) to get Hermes deployed in their cloud (**M1, M4**, see **R4**). Initially, I will prototype the serverless service on the VU-DAS clusters and seek feedback from the Dutch user-community. Then, in collaboration with Mellanox and the HPC-AI Advisory Council, I will test the D3G API on their state-of-the-art infrastructure. I will leverage MCS's close collaboration with Databricks (company behind Spark) to deploy Hermes in their cloud-settings, and seek their feedback. I will also reach out to IBM to understand their clients' requirements, and co-develop user-facing data processing services by leveraging state-of-the-art hardware in their cloud (POWER9, NVlink). IBM Research actively engages researchers from academia to have joint research collaborations (I was beneficiary of such an outreach as they funded my PhD research).

(c) **Academic peer-reviewed publications for computer scientists:** Extending our understanding of how to build high-performance software in Post-Moore's Law world, I will publish in top-tier peer-reviewed conferences (**R1**, **M5**):

(i) The design and implementation of D3G API to capture the dataflow among devices in a distributed system. Two workloads that I will explore are distributed sorting (MapReduce) [74] and DawnBench (Deep learning) [75], both of which are public competitions and thus reproducible (**M2-3**).

(ii) The architecture of a Hermes-accelerated data-processing framework. I will discuss techniques to automatically synthesize dataflows and map them to devices, using D3G at runtime. I will demonstrate how these techniques can be extended beyond Spark to other data-processing frameworks (**M2-3**).

(iii) A study of how modern devices are used inside data-processing frameworks, including device utilization and usage-cost, based on a statistical analysis of user-workload traces (**M1,M4**).

(iv) A serverless data-processing architecture to transparently manage new accelerator-devices by tracking and storing their states in a distributed, shared store (**M1-M4**).

2c. Number of words used

Section 2a: 1995 words (max. 2,000 words)

Section 2b: 720 words (max. 750 words)

2d. Literature references

1. Thomas N. Theis and H. -S. Philip Wong, *"The End of Moore's Law: A New Beginning for Information Technology"*, IEEE Computing in Science and Engg. 19, 2 (March 2017), pages 41-50.
2. Mihir Nanavati, Malte Schwarzkopf, Jake Wires, and Andrew Warfield, *"Non-volatile Storage"*, Queue 13, 9, Pages 20 (November 2015), 24 pages.
3. David Patterson and John L. Hennessy, *"A New Golden Age for Computer Architecture: Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development"*, Turing Award lecture, https://www.acm.org/hennessy-patterson-turing-lecture, 2018.
4. Google Cloud, *"Cloud TPU: Train and run machine learning models faster than ever before"*, https://cloud.google.com/tpu/, 2018.
5. Google Cloud, *"Graphic Processing Unit (GPU), Leverage GPUs on Google Cloud for machine learning, scientific computing, and 3D visualization"*, https://cloud.google.com/gpu/, 2018.

6. LiquidIO II Smart NICs, https://www.marvell.com/ethernet-adapters-and-controllers/liquidio-smart-nics/, 2018.
7. Mellanox, BlueField SmartNIC Ethernet, https://www.mellanox.com/page/products_dyn?product_family=275&mtag=bluefield_smart_nic&ssn=9el8l3sv4jie8setg8q8mdi0n2, 2018.
8. Daniel Firestone, and others, *"Azure Accelerated Networking: SmartNICs in the Public Cloud"*, 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), pages 51-66, 2018.
9. Philip Kufeldt, Carlos Maltzahn, Tim Feldman, Christine Green, Grant Mackey, and Shingo Tanaka, *"Eusocial Storage Devices: Offloading Data Management to Storage Devices that Can Act Collectively",* USENIX ;login:, summer 2018, vol. 43, no. 2.
10. Jae Do, *"SoftFlash: Programmable Storage in Future Data Centers",* https://www.snia.org/sites/default/files/SDC/2017/presentations/Storage_Architecture/Do_Jae_Young_SoftFlash_Programmable_Storage_in_Future_Data_Centers.pdf, SNIA SDC, Santa Clara, CA, 2017.
11. Amazon EC2, *"F1 Instances, Run Customizable FPGAs in the AWS Cloud",* https://aws.amazon.com/ec2/instance-types/f1/, 2018.
12. Mohamed Zahran, *"Heterogeneous Computing: Here to Stay",* Queue 14, 6, Pages 40 (December 2016).
13. Animesh Trivedi, Patrick Stuedi, Jonas Pfefferle, Radu Stoica, Bernard Metzler, Ioannis Koltsidas, and Nikolas Ioannou, *"On the [ir]relevance of network performance for data processing"*, Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing (HotCloud'16). USENIX Association, USA, pages 126-131.
14. Animesh Trivedi, Patrick Stuedi, Jonas Pfefferle, Adrian Schuepbach, Bernard Metzler, "*Albis: High-Performance File Format for Big Data Systems*", Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC 18), Boston, MA, USA, July 2018.
15. Animesh Trivedi, Nikolas Ioannou, Bernard Metzler, Patrick Stuedi, Jonas Pfefferle, Kornilios Kourtis, Ioannis Koltsidas, and Thomas R. Gross, *"FlashNet: Flash/Network Stack Co-Design",* ACM Trans. Storage 14, 4, Article 30 (December 2018).
16. Matias Bjørling, Jens Axboe, David Nellans, and Philippe Bonnet, *"Linux block IO: introducing multi-queue SSD access on multi-core systems"*, Proceedings of the 6th International Systems and Storage Conference (SYSTOR '13). ACM, New York, NY, USA, Article 22 , 10 pages.
17. Sean Hefty, *"Rsockets"*, OpenFabrics International Workshop, Monterey, CA, USA, 2012.
18. Mahidhar Tatineni, Xiaoyi Lu, Dongju Choi, Amit Majumdar, and Dhabaleswar K. (DK) Panda, *"Experiences and Benefits of Running RDMA Hadoop and Spark on SDSC Comet",* Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale (XSEDE16). ACM, New York, NY, USA, Article 23, 5 pages.
19. Xiaoyi Lu, Dipti Shankar, Shashank Gugnani, and Dhabaleswar K. (DK) Panda, *"High-Performance Design of Apache Spark with RDMA and Its Benefits on Various Workloads"*, IEEE Big Data (Big Data), 2016.
20. Apache Crail blog post, *"Sorting on a 100Gbit/s Cluster using Spark/Crail"*, http://crail.incubator.apache.org/blog/2017/01/sorting.html, 2018.
21. Anuj Kalia, Michael Kaminsky, and David G. Andersen, *"Using RDMA efficiently for key-value services",* Proceedings of the 2014 ACM conference on SIGCOMM (SIGCOMM '14). ACM, New York, NY, USA, 295-306.
22. John Ousterhout, Arjun Gopalan, Ashish Gupta, Ankita Kejriwal, Collin Lee, Behnam Montazeri, Diego Ongaro, Seo Jin Park, Henry Qin, Mendel Rosenblum, Stephen Rumble, Ryan Stutsman, and Stephen Yang, *"The RAMCloud Storage System",* ACM Trans. Comput. Syst. 33, 3, Article 7 (August 2015), 55 pages.
23. Aleksandar Dragojević, Dushyanth Narayanan, Orion Hodson, and Miguel Castro, *"FaRM: fast remote memory",* Proceedings of the 11th USENIX NSDI, pages 401-414.
24. Patrick Stuedi, Animesh Trivedi, and Bernard Metzler, *"Wimpy nodes with 10GbE: leveraging one-sided*

operations in soft-RDMA to boost memcached", Proceedings of the 2012 USENIX conference on Annual Technical Conference (USENIX ATC'12). USENIX Association, Berkeley, CA, USA.

25. Animesh Trivedi, Patrick Stuedi, Bernard Metzler, Clemens Lutz, Martin Schmatz, Thomas R. Gross, *"RStore: A Direct-Access DRAM-based Data Store"*, Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS'15), Columbus, Ohio, USA, 2015.
26. Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle, Radu Stoica, Bernard Metzler, Nikolas Ioannou, Ioannis Koltsidas, *"Crail: A High-Performance I/O Architecture for Distributed Data Processing"*, IEEE Bulletin of the Technical Committee on Data Engineering, Special Issue on Distributed Data Management with RDMA, Volume 40, pages 40-52, March, 2017.
27. Anuj Kalia, Michael Kaminsky, and David G. Andersen, *"FaSST: fast, scalable and simple distributed transactions with two-sided (RDMA) datagram RPCs",* Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'16). USENIX Association, pages 185-201, 2016.
28. Xingda Wei, Zhiyuan Dong, Rong Chen, Haibo Chen, *"Deconstructing RDMA-enabled Distributed Transactions: Hybrid is Better!",* Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation (OSDI'18). USENIX Association, Berkeley, CA, USA, 233-251.
29. Patrick Stuedi, Animesh Trivedi, Bernard Metzler, and Jonas Pfefferle. *"DaRPC: Data Center RPC"*, Proceedings of the ACM Symposium on Cloud Computing (SOCC '14), Article 15 , 13 pages.
30. Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe, *"Arrakis: The Operating System Is the Control Plane",* ACM Trans. Comput. Syst. 33, 4, Article 11 (November 2015), 30 pages.
31. Phitchaya Mangpo Phothilimthana, Ming Liu, Antoine Kaufmann, Simon Peter, Rastislav Bodik, Thomas Anderson, *"Floem: A Programming System for NIC-Accelerated Network Applications"*, Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'18), USA, pages 663-679, 2018.
32. Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, Arvind Krishnamurthy, *"TVM: An Automated End-to-End Optimizing Compiler for Deep Learning"*, Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation (OSDI'18). USENIX Association,USA, pages 578-594, 2018.
33. Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar. *"Advances in dataflow programming languages",* ACM Comput. Surv. 36, 1 (March 2004), 1-34, 2014.
34. A. L. Davis and R. M. Keller, *"Data Flow Program Graphs",* IEEE Computer 15, 2 (February 1982), pages 26-41, 1982.
35. Naim A. Kheir, *"Systems Modeling and Computer Simulation",* (2nd ed.). Marcel Dekker, Inc., NY, USA.
36. Yair Levy, Timothy J. Ellis, "*A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research",* Informing Sci J 9: 181-212 (2006).
37. R. Hamming, *"The Art of Doing Science and Engineering: Learning to Learn"*, CRC Press, 1997.
38. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, *"A Design Science Research Methodology for Information Systems Research"*, Journal of Management Information Systems 24(3): 45-77 (2008).
39. R. Jain, *"The Art of Computer Systems Performance Analysis"*, John Wiley & Sons Inc., New York, USA, 1991.
40. Gernot Heiser, *"Systems Benchmarking Crimes"*, http://www.cse.unsw.edu.au/~Gernot/benchmarking-crimes.html, 2018.
41. S. Shen, V. van Beek, and A. Iosup, *"Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters"*, CCGRID 2015: 465-474
42. A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, D. H. J. Epema, *"The Grid Workloads Archive"*, Future Generation Comp. Syst. 24(7): 672-686 (2008).
43. D. Kondo, B. Javadi, A. Iosup, D. H. J. Epema, *"The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems"*, CCGRID 2010: 398-407.

44. B. Zhang, A. Iosup, J. Pouwelse, D. H. J. Epema, *"The peer-to-peer trace archive: design and comparative trace analysis"*, Proceedings of the ACM CoNEXT Student Workshop (CoNEXT '10 Student Workshop). ACM, New York, NY, USA, Article 21.
45. Sonja Bezjak, April Clyburne-Sherin, Philipp Conzett, Pedro Fernandes, Edit Görögh, Kerstin Helbig, Bianca Kramer, Ignasi Labastida, Kyle Niemeyer, Fotis Psomopoulos, Tony Ross-Hellauer, René Schneider, Jon Tennant, Ellen Verbakel, Helene Brinken, Lambert Heller, *"Open Science Training Handbook",* Zenodo, DOI: https://doi.org/10.5281/zenodo.1212496, April, 2018.
46. Kay Ousterhout, Ryan Rasti, Sylvia Ratnasamy, Scott Shenker, and Byung-Gon Chun, *"Making sense of performance in data analytics frameworks",* Proceedings of the 12th USENIX NSDI, USA, pages 293-307.
47. Ana Klimovic, Heiner Litz, Christos Kozyrakis, *"Selecta: Heterogeneous Cloud Storage Configuration for Data Analytics"*, Proceedings of the USENIX Annual Technical Conference (ATC), Boston, MA, July 2018.
48. Erwin van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uta, Alexandru Iosup, *"Serverless is More: From PaaS to Present Cloud Computing"*, IEEE Internet Computing, Sep/Oct edition, 2018.
49. Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht, *"Occupy the cloud: distributed computing for the 99%",* Proceedings of the ACM Symposium on Cloud Computing (SoCC '17), 2017.
50. Ana Klimovic, Yawen Wang, Patrick Stuedi, Animesh Trivedi, Jonas Pfefferle and Christos Kozyrakis, *"Pocket: Ephemeral Storage for Serverless Analytics",* Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI'18), 2018.
51. Ana Klimovic, Yawen Wang, Christos Kozyrakis, Patrick Stuedi, Jonas Pfefferle, and Animesh Trivedi, *"Navigating Storage for Serverless Analytics"*, Proceedings of the 2018 USENIX Annual Technical Conference (ATC), 2018.
52. Animesh Trivedi, Patrick Stuedi, Bernard Metzler, Roman Pletka, Blake G. Fitch, Thomas R. Gross, "*Unified High-Performance I/O: One Stack to Rule Them All*", Proceedings of the 14th Workshop on Hot Topics in Operating Systems (HotOS XIV), Santa Ana Pueblo, NM, USA, May 2013.
53. Patrick Stuedi, Bernard Metzler, and Animesh Trivedi, *"jVerbs: ultra-low latency for data center applications"*. In Proceedings of the 4th annual Symposium on Cloud Computing (SOCC '13). ACM, New York, NY, USA, Article 10, 14 pages.
54. Animesh Trivedi, Bernard Metzler, Patrick Stuedi, *"A Case for RDMA in Clouds: Turning Supercomputer Networking into Commodity",* Proceedings of the 2nd ACM SIGOPS APSys, China, July 2011.
55. Animesh Trivedi, "*Data processing at the speed of 100 Gbps@Apache Crail (Incubating)"*, https://dataworkssummit.com/san-jose-2018/session/data-processing-at-the-speed-of-100-gbpsapache-crail-incubating/, DataWorks Summit talk, June, 2018.
56. Github, Software artifacts (including Spark shuffler, broadcast, Albis, DisNI, DaRPC, SoftiWARP) from the High-performance I/O Research Group at IBM Research Lab, Zurich, https://github.com/zrlio, 2018.
57. Shai Bergman, Tanya Brokhman, Tzachi Cohen, and Mark Silberstein, *"SPIN: seamless operating system integration of peer-to-peer DMA between SSDs and GPUs",* Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC '17). USENIX Association, Berkeley, CA, USA, 167-179.
58. Gilad Shainer, Ali Ayoub, Pak Lui, Tong Liu, Michael Kagan, Christian R. Trott, Greg Scantlen, and Paul S. Crozier, "*The development of Mellanox/NVIDIA GPUDirect over InfiniBand - A new model for GPU to GPU communications",* Comput. Sci. 26, 3-4 (June 2011), pages 267-273.
59. Apache Spark, Lightning-fast unified analytics engine, https://spark.apache.org/, 2018.
60. Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica, "*Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing*", Proceedings of the 9th USENIX conference on