Reconfigurable Data Processing for Clouds

Anil Madhavapeddy
Computer Laboratory
University of Cambridge
15 JJ Thomson Avenue, Cambridge
avsm2@cl.cam.ac.uk

Satnam Singh
Roger Needham Building
Microsoft Research
7 JJ Thomson Ave, Cambridge
satnams@microsoft.com

***Abstract*—Reconfigurable computing in the cloud helps to solve many practical problems relating to scaling out data-centers where computation is limited by energy consumption or latency. However, for reconfigurable computing in the cloud to become practical several research challenges have to be addressed. This paper identifies some of the perquisites for reconfigurable computing systems in the cloud and picks out several scenarios made possible with immense cloud-based computing capability.**

***Keywords*-reconfigurable computing; cloud computing.**

I. Introduction

Reconfigurable computing for some time has had the potential to make a huge impact on mainstream high performance computing. We now have very large capacity FPGAs which contain many highly parallel fine grain parallel processing power, and the ability to define high bandwidth custom memory hierarchies offers a compelling combination of flexibility and performance. However, mainstream adoption of reconfigurable computing has been hampered by the need to use and maintain specialized FPGA-based boards and clusters and the lack of programming models that make this technology accessible to regular programmers. FPGAs do not enjoy first class operating system support and lack the application binary interfaces (ABIs) and abstraction layers that other co-processing technologies enjoy (most notably GPUs).

We believe it is time to place FPGAs on the same blades as GPUs and CPUs in the cloud and offer them as a managed service with a high-level programming interface. We see a new dawn for reconfigurable computing that makes this exciting technology available to millions of developers without taking on the burden of maintaining specialized hardware, and without having to invest in complex tool-chains and programming models based on the low-level details of circuit design. This paper explores cloud-based heterogeneous computing and identifies some requirements needed to make it a reality for a wider class of developers.

The major limitation on the growth potential of data-centres is now energy consumption, and it is here that specialised computing resources like FPGAs can have significant impact: allowing us to scale cloud operations to an extent which inefficient using just conventional processors. The availability of very large scale reconfigurable computing devices enables the deployment of this flexible technology in contexts where in the past capacity limitations have prevented their use as a generalized, shared and virtualized resource.

A. Cloud Computing

Just a decade ago, it was common practise to purchase physical machines and place them with a hosting company. As the Internet's popularity grew, their reliability and availability requirements also grew beyond a single data-center. Sites such as Google and Amazon started building huge silos in the USA and Europe, with correspondingly larger energy demands. These providers had to provision for their peak load, and had much idle capacity at other times.

At the same time, researchers were examining the possibility of dividing up commodity hardware into isolated chunks of computation and storage, which could be rented on-demand. The Xen hypervisor was developed as an open-source solution to partition multiple untrusted operating systems [1], and subsequently adopted by Amazon to underpin its Elastic Computing service. It became the first commercial provider of "cloud computing"—renting a slice of a data-center to provide on-demand resources that can be dynamically scaled up and down according to demand.

Cloud computing brought reconfigurable computing to the software arena. Hardware resources are now dynamic, and so sudden surges in load can be adapted to by adding more virtual machines to a server pool until it subsides. This resulted in a surge of new datacenter components designed to scale across machines, ranging from storage systems like Dynamo [2] to distributed computation with MapReduce [3].

B. Where are the Hardware Clouds?

Data-centers encourage horizontal scaling by increasing the number of hosts. The vertical scaling model (more powerful individual machines) is difficult due to the shift to multi-core CPUs with fairly constant clock speed. IBM notes that with "*data-centers using 10–30 times more energy per square foot than office space, energy use doubling every 5 years, and [..] delayed capital investments in new power plants*", something must change [4]. The growth potential of

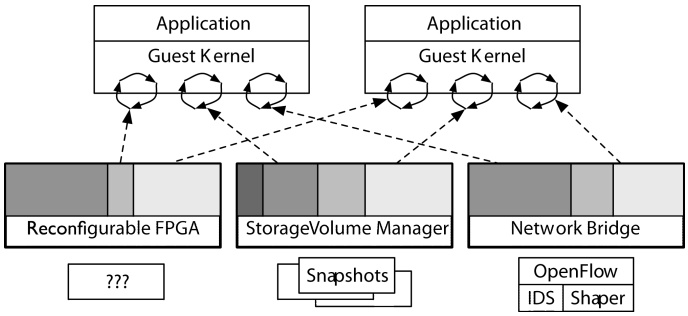

Figure 1. Split-trust devices on virtualised platforms have a management domain that partitions physical resources, allocates portions to guests, and enforces access rights. The details of the management policy is specific to the type of resource, such as storage or networking.

these data-centers is now energy limited, and the inefficiency of the software stack is beginning to take its toll.

II. Research Directions

Cloud computing is quite new and evolving. We now look at some of the major interest areas from that community, and examine how reconfigurable FPGAs could help.

A. Operating Systems

The traditional role of an operating system of partitioning physical hardware is quite different when virtualised. Hypervisors expose simple network and storage interfaces to virtual machines (VMs), with the actual physical drivers handled elsewhere [5]. Kernels that only run as VMs need just a few device drivers to work, and no longer have to support the full spectrum of devices. Figure 1 illustrates the difference between *managing* physical devices and *using* a portion of that resource from a virtualised application.

The management APIs in the control domain differ across resource types. Networking involves bridging and topology and integration with systems such as OpenFlow [6]. Storage is concerned with snapshots and de-duplication of common blocks across VMs [7]. However, the device exposed to the VMs remains simple; often little more than a shared-memory page with consumer/producer channel. These devices are currently used for I/O, but could be extended to actually run computation over the data, with the same high-throughput and low-latency requirements. The VM could specify the computation (e.g. as a DSL), and the management tools interface it with the physical board and manage sharing across VMs. This is simple from the programmer's perspective, and portable across different FPGA boards and tool-chains.

The availability of GPUs and programmable I/O boards have led to the development of new software architectures. Helios is a new operating system designed to simplify the task of writing, deploying and profiling code across heterogeneous platforms [8]. It introduces "satellite kernels" that export a uniform set of OS abstractions, but are independent tasks that run across different resources.

B. Datacenter Programming

Processing large datasets requires efficiently partitioning the computation across many hosts. Distributed dataflow frameworks such as MapReduce [3], Dryad [9] and CIEL [10] all expose a simple programming model and transparently handle the difficult aspects of distribution: fault tolerance, resource scheduling, synchronization and message passing.

These frameworks all build Directed Acyclic Graphs (DAGs), where the nodes represent data and the edges are computation over the data. The run-time schedules compute on hosts and iteratively walks the DAG until a result is obtained. It also prepares the host to ensure required data is available locally. This preparation step can also include compilation, and so an FPGA DSL could be transparently scheduled to hardware (as available) or executed in software if not available. The main challenge is to track the cost of reconfiguring FPGAs rather than just executing it in software, but this is made easier since the run-time can inspect the size of the input data at runtime. Mesos [11] investigates how to partition physical resources across multiple competing frameworks operating on the same set of hosts, which is useful when considering fixed-size FPGA boards.

The recent surge of new components designed specifically for datacenters also encourages research into new database models that depart from SQL and traditional ACID models. Mueller *et al.* programmed data processing operators on top of large FPGAs, and concluded that the right computation model is essential (e.g. an asynchronous sorting network) [12]. Within these constraints however, they had comparable performance and significantly improved power-consumption and parallellisation—both areas essential to successful datacenter databases in the modern world.

A close integration between high-level host languages and FPGAs will greatly help adoption by mainstream programmers. The fact that C and C++ are considered low-level languages in the cloud, and high-level to FPGA programmers is indicative of the cultural difference between the two communities! There are a number of promising efforts that embed DSLs in C/C++ code to ease their integration. MORA is a DSL for streaming vector and matrix operations, aimed at multimedia applications [13]. Designs can be compiled into normal executables for functional testing, before being retargeted at a hardware array. MARC uses the LLVM compiler infrastructure to convert C/C++ code to FPGAs [14]. Although performance is still lower than a manually optimised FPGA implementation, it is significantly less effort to design and implement portably due to its higher-level approach. This quicker code/deployment/results cycle is essential to incrementally get feedback about code for the more casual programmer, who is using rented cloud computing resources in order to save time in the first place.

Some languages now separate data-parallel processing

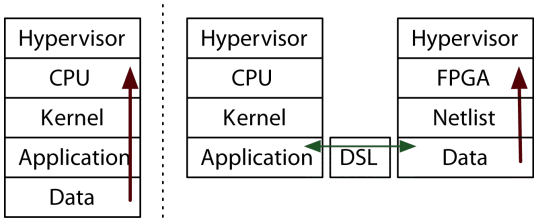

Figure 2. Malicious data can be crafted to exploit memory errors and execute as code on a CPU *(left)*. With an FPGA interface, it cannot execute arbitrary code, and the CPU never iterates over the data *(right)*.

explicitly so that they can utilise resources such as GPUs. Data parallel Haskell integrates the full range of types available modern languages, and allows sum types, recursive types, higher-order functions and separate compilation [15]. Accelerator is a library to synthesise data-parallel programs written in C# directly to FPGAs [16]. A more radical embedding is via multi-stage programming, where programmers specify abstract algorithms in a high-level language that is put through a series of translation stages into the desired architecture [17]. All of these approaches are highly relevant to reconfigurable FPGA computing in the cloud, as they extend existing, familiar programming languages with the constraints required to compile sub-sets into hardware.

C. Information Security

The cloud is often used to outsource processing over large datasets. The code implementing the batch-processing is often written in C or Fortran, and a bug in handling input data can let attackers execute arbitrary code on the host machine (see Figure 2, *left*). Although the hypervisor layer contains the attacker inside the virtual machine, they still have access to many of the local network resources, and worse, other (possibly sensitive) datasets. Exploits are mitigated by using software privilege separation, but this places trust in the OS kernel layer instead [18].

Data processing on the cloud using reconfigurable FPGAs offers an exceptional improvement in security by shifting that trust from software to hardware. Baking algorithm implementations into FPGAs entirely removes the capability of attackers to run arbitrary code, thus enforcing strong privilege separation. The application compiles its algorithms to an FPGA, and never directly manipulates the data itself via the CPU. Malicious data never gets the opportunity to run on the host CPU, and instead only a small channel exists between the OS and FPGA to communicate results (see Figure 2, *right*). The conventional threat model for FPGAs is that a physical attacker can compromise its hardware SRAM. When deployed in the cloud, the attacker cannot gain physical access, leaving few attack vectors.

Moving beyond low-level security, there is also a realisation that data contents needs protection against untrusted cloud infrastructure [19]. Encoding data processing tasks across FPGAs enforces a data-centric view of computation, distinct from coordinating computation (e.g. load balancing or fault tolerance, which cannot compromise the contents of data). Programming language researchers have mechanisms for encoding information flow constraints [20], and more recently, statistical privacy properties [21]. These techniques are often too intrusive to fully integrate into general-purpose languages, but are ideal for the domain-specific data-flow languages which provide the interface between general-purpose and reconfigurable FPGA computing.

Another intriguing development is homomorphic encryption, which permits computation over encrypted data without being able to ever decrypt the underlying data. The utility of homomorphic encryption has been recognised for decades, but has so far been extremely expensive to implement [22]. Cloud computing revitalises the problem, as malicious providers might be secretly recording data or manipulating results. Recently, there have been several lattice-based cryptography schemes that reduce the complexity cost of homomorphic encryption [23]. Lattice reduction can be significantly accelerated via FPGAs; Detrey *et al.* report a speedup of 2.12 of an FPGA versus a multi-core CPU of comparable costs [24]. This points to a future where reconfigurable million-LUT FPGAs could be used to perform computation where even the cloud vendor is untrusted!

Reducing the cost of cryptography in the cloud could also have significant social impact. The Internet has seen large-scale deployment of anonymity networks such as Tor [25] and FreeNet for storing data [26]. Due to the encryption requirements imposed by onion-routing, access to such networks remains slow and high-latency. There have been proposals to shift the burden of anonymous routing into the cloud to fix this, but reducing the cost (financially) remains one of the key barriers to more widespread adoption of anonymity [27]. This is symptomatic of the broader problem of improving networking performance in the cloud. Central control systems such as OpenFlow are rapidly gaining traction, along with high-performance implementation in hardware [28]. Virtual networking is reconfigured much more often than hardware setups (e.g. for load-balancing or fault tolerance), and services such as Tor further increase the gate requirements as computation complexity increases [29].

The challenge, then, for integration into the cloud, is how to unify the demands of data-centric processing, language integration, network processing into a single infrastructure. Specific problems that have to be addressed in addition to those mentioned earlier include:

- The need for better OS integration, device models, and abstractions (as with split-trust in Xen described earlier)
- Without an ABI or source API, software re-use and integration is very difficult. How can (for example) OpenSSL transparently take advantage of an FPGA?
- Debugging and visualization support. General pur-

pose systems provide a hypervisor-kernel-userspace-language runtime model that gets progressively easier and higher-level to debug. Abstraction boundaries exist where they don't in current FPGAs. Staged programming or functional testing in a general-purpose systems makes this easier.

- We need to develop a common set of concepts, principles and models for application execution on reconfigurable computing platforms to allow collaboration between universities and companies and to provide a solid framework to build new innovations and applications. This kind of eco-system has been sadly lacking for reconfigurable systems.

It is encouraging that cloud computing is driven by fine-grained charging for the computation resources used. Reconfigurable FPGAs driven down the cost of many types of computation commonly found on the cloud, and thus a community-driven deployment of a cloud setup with rentable hardware would provide a focal point to "fill in the blanks" for reconfigurable FPGA computing in the cloud.

It is not clear that these goals can be achieved by a collection of parallel independent university and industry projects. What is needed is a coordinated research program involving members of the reconfigurable computing community working with each other and researchers in cloud computing to define a new vision of where we would like to go and then set standards etc. to try and achieve that goal. For this to happen we will need some kind of wide ranging joint research project proposal or standardization effort.

III. Conclusion

Reconfigurable computing is at the cusp of rising up from being a niche activity accessible to only a small group of experts to becoming a mainstream computing fabric used in concert with other heterogeneous computing elements like GPUs. For this to become a reality we need to combine some of the successes in the FPGA-based research with new thinking about programming models to create a development environment for 'civilian programmers'. This will require collaboration between researchers in architecture, CAD tools, programming languages and types, run-time system development, web services, scripting and orchestration, re-targetable compilation, instrumentation and monitoring of heterogeneous systems, and failure management. Furthermore, the requirements of reconfigurable computing in a shared cloud service context also places new requirements on CAD tools and architectures which are at odds with their current requirements. Today FPGA vendors produce architectures for use in an embedded context to be programmed by digital design engineers.

Yesterday's programmers of reconfigurable systems were highly trained digital designers using Verilog. Today we are at the cusp of a revolution which will make tomorrow's users of reconfigurable technology from regular software engineers who map their algorithms onto a heterogeneous mixture of computing resources to achieve currently unachievable levels of performance, management of energy consumption and the execution of scenarios which promise an ever more interconnected world. This paper has set out a vision for a reconfigurable computing system in the cloud, identified important research challenges and promising research directions and illustrated scenarios that are made possible by reconfigurable computing in the cloud.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*. New York, NY, USA: ACM, 2003, pp. 164–177.

[2] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *Proceedings of ACM SIGOPS Symposium on Operating Systems Principles*, 2007, pp. 205–220.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in *Proceedings of the 6th Conference on Operating Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[4] J. B. Carter, "A Look Inside IBM's Green Data Center Research," in *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '09. ACM, 2009, pp. 153–154.

[5] A. Warfield, S. Hand, K. Fraser, and T. Deegan, "Facilitating the Development of Soft Devices," in *Proceedings of the USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 22–22.

[6] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, March 2008.

[7] D. T. Meyer, G. Aggarwal, B. Cully, G. Lefebvre, M. J. Feeley, N. C. Hutchinson, and A. Warfield, "Parallax: virtual disks for virtual machines," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 41–54, April 2008.

[8] E. B. Nightingale, O. Hodson, R. McIlroy, C. Hawblitzel, and G. Hunt, "Helios: Heterogeneous Multiprocessing with Satellite Kernels," in *Proceedings of the 22nd Symposium on Operating Systems Principles*. New York, NY, USA: ACM, 2009, pp. 221–234.

[9] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 59–72, March 2007.

[10] D. G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand, "CIEL: a universal execution engine for distributed data-flow computing," in *Proceedings of the 8th USENIX Symposium on Networked System Design and Implementation (NSDI)*. USENIX, 2011.

[11] B. Hindman, A. Konwinski, M. Zaharia, and I. Stoica, "A common substrate for cluster computing," in *Proceedings of the 2009 conference on Hot Topics in Cloud Computing*, ser. HotCloud'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 19–19.

[12] R. Mueller, J. Teubner, and G. Alonso, "Data processing on FPGAs," *Proc. VLDB Endow.*, vol. 2, pp. 910–921, August 2009.

[13] W. Vanderbauwhede, M. Margala, S. R. Chalamalasetti, and S. Purohit, "A C/C++-embedded domain-specific language for programming the mora soft processor array," in *21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP)*, Jul. 2010, pp. 141 –148.

[14] I. Lebedev, S. Cheng, A. Doupnik, J. Martin, C. Fletcher, D. Burke, M. Lin, and J. Wawrzynek, "Rethinking FPGA computing with a many-core approach," in *International Conference on Reconfigurable Computing and FPGAs*, Dec. 2010.

[15] M. M. T. Chakravarty and G. Keller, "More types for nested data parallel programming," in *Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, ser. ICFP '00. New York, NY, USA: ACM, 2000, pp. 94–105.

[16] S. Singh, "Integrating FPGAs in high-performance computing: programming models for parallel systems – the programmer's perspective," in *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays*, ser. FPGA '07. New York, NY, USA: ACM, 2007, pp. 133–135.

[17] F. Chen, R. Goyal, E. Westbrook, and W. Taha, "Implicitly heterogeneous multi-stage programmng for FPGAs," in *Proceedings of the Eleventh Symposium on Trends in Functional Programming (TFP)*, May 2010.

[18] N. Provos, M. Friedl, and P. Honeyman, "Preventing privilege escalation," in *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*. Berkeley, CA, USA: USENIX Association, 2003, pp. 16–16.

[19] W. Zhou, M. Sherr, W. R. Marczak, Z. Zhang, T. Tao, B. T. Loo, and I. Lee, "Towards a data-centric view of cloud security," in *Proceedings of the second international workshop on Cloud data management*, ser. CloudDB '10. New York, NY, USA: ACM, 2010, pp. 25–32.

[20] A. Sabelfeld and A. Myers, "Language-based information-flow security," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 1, pp. 5 – 19, Jan. 2003.

[21] J. Reed and B. C. Pierce, "Distance makes the types grow stronger: a calculus for differential privacy," in *Proceedings of the 15th ACM SIGPLAN international conference on Functional programming*, ser. ICFP '10. New York, NY, USA: ACM, 2010, pp. 157–168.

[22] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, pp. 169–177, 1978.

[23] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178.

[24] J. Detrey, G. Hanrot, X. Pujol, and D. Stehlé, "Accelerating lattice reduction with FPGAs," in *Proceedings of the First international conference on Progress in cryptology: cryptology and information security in Latin America*, ser. LATINCRYPT'10. Springer-Verlag, 2010, pp. 124–143.

[25] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 21–21.

[26] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: a distributed anonymous information storage and retrieval system," in *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*. New York, NY, USA: Springer-Verlag New York, Inc., 2001, pp. 46–66.

[27] R. Mortier, A. Madhavapeddy, T. Hong, D. Murray, and M. Schwarzkopf, "Using dust clouds to enhance anonymous communication," in *Eighteenth International Workshop on Security Protocols*, Apr. 2010.

[28] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '08. New York, NY, USA: ACM, 2008, pp. 1–9.

[29] D. Unnikrishnan, R. Vadlamani, Y. Liao, A. Dwaraki, J. Crenne, L. Gao, and R. Tessier, "Scalable network virtualization using fpgas," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, ser. FPGA '10. New York, NY, USA: ACM, 2010, pp. 219–228.