The Multihype: Virtualizing Heterogeneous-ISA Architectures

Pierre Olivier[1], Binoy Ravindran[1], Antonio Barbalace[2]
[1]Virginia Tech, [2]Stevens Institute of Technology
polivier@vt.edu, binoy@vt.edu, abarbala@stevens.edu

1 Introduction

System-level virtualization enables, amongst other benefits, consolidation and multi-tenancy in a secure way [8]. As one of the technologies powering cloud computing, it is broadly adopted in the datacenter. However, with the cloud extending to the edge, and the recent acceptance of system-level virtualization in portable and control systems to consolidate real-time and not real-time software, virtualization technologies are expanding into all computer market segments.

Modern computer architectures are built with an increasing number of heterogeneous processors. They include accelerators (GPU [37], TPUs [23]), re-configurable FPGAs [17, 36], as well as general-purpose processors. The latter category embraces current machines with processors of the same ISA but different ISA extensions or microarchitecture (single-ISA heterogeneity), such as ARM big.LITTLE [10], Xeon/Xeon Phi [9], or Intel Sunny Cove [21]. Moreover, platforms with general-purpose processors of completely different ISAs are becoming more popular: AMD is integrating ARM cores in its x86 processors line, and intelligent network cards/storage devices powered by ARM [16, 31], MIPS64 [29] or Tile [30] cores turn an x86 machine into a hybrid ARM/MIPS/Tile-x86. Finally, SoC combining multiple fully heterogeneous general-purpose processors on cache coherency have been simulated [40], and prototyped [26]. Although the latter includes RISC-V and SPARC, the new Intel Skylake with FPGA [22] opens the possibility to prototype RISC-V and x86 – both including hardware support for virtualization.

Contribution. Considering the wide adoption of virtualization and the growing interest in heterogeneous processing, in this paper we try to answer the question: *how should platforms with heterogeneous processors be virtualized?* In fact, the systems software community proposed OSes designs able to fully exploit such heterogeneous platforms [1, 2, 4, 27]. However, to the best of our knowledge there is no work on system-level virtualization in that domain. Bringing virtualization to these platforms would enable traditional benefits such as consolidation, fault-tolerance, and secure multi-tenancy, etc. Moreover, it would also make such heterogeneous setups more accessible through cloud/edge deployments. This solution could also potentially enable new models of resource pricing for cloud providers.

This paper focuses on heterogeneous systems built with general-purpose (OS-capable) processors and aims at identifying the challenges of virtualizing them. We target single-ISA heterogeneous platforms, as well as platforms with processors of completely different ISAs. We observe the success of the multiple-kernel OS design [1, 2, 4, 27] in managing heterogeneous platforms and reason about porting that model to the virtualization world. We introduce the *Multihype*, an hypervisor for platforms with heterogeneous processors.

2 Motivation & Background

Why Virtualize Heterogeneous-ISA Architectures? Even for current OS-capable heterogeneous systems implementing a single ISA but with different extensions and microarchitecture features, virtualization support is far from ideal. For example, ARM big.LITTLE [10] is poorly supported by both Xen [3] or KVM [24] hypervisors [15, 41]. Because of microarchitectural differences (e.g., cache line sizes), virtual CPUs cannot migrate between diverse processors (e.g., big and little). This limits the potential of this technology by confining its usage among identical CPUs only.

Although multiple studies targeted the virtualization of existing CPU/GPU heterogeneous systems [14, 18, 20, 34, 38], we are unaware of any effort trying to virtualize OS-capable heterogeneous-ISA platforms. Virtualizing such platforms is fundamentally distinct from CPU/GPU virtualization due to the differences in the way GPUs and CPUs are programmed and interfaced. Virtualizing GPUs and other accelerators falls within the domain of device virtualization. There is a specific interface/protocol between the GPU and the CPU that needs to be virtualized or emulated. This is very different from the software/hardware interface between the guest OS and the VM that needs to be virtualized in the case of system-level virtualization. Because of these differences, this work focuses on OS-capable platforms and scopes out CPU/GPU setups.

Heterogeneity Scenarios. By virtualizing heterogeneous platforms comes the possibility of running multiple potentially heterogeneous virtual machines (VMs) on a single heterogeneous physical machine. As mentioned in the introduction, enterprise servers may pair x86 CPUs together with ARM, MIPS, etc. A heterogeneous virtual machine enables a cloud provider to share between multiple tenants all such processors – not just the x86, and providers may still apply the classical pay-as-you-go pricing model of the cloud. This will foster the democratization of heterogeneous-ISA hardware, amplify the availability, affordability, performance, of the virtual machines in the cloud, and increase the return-on-investment for the cloud provider.

Several studies [1, 35, 40] demonstrated the benefits of heterogeneous-ISA platforms in terms of performance, or power consumption at different integration scale. For example, by enabling the software to runtime migrate between heterogeneous cores for applications potentially exhibiting multiple phases [35, 40], where the affinity goes towards a different ISA per phase. It is foreseen that heterogeneous-ISA machines in the cloud (and cloud-edge) will introduce a new flexible resource offer and pricing model. For example, in elastic scenarios where reactivity is critical [33], a user may decide to rent a VM on slow and cheap cores. However, in case of a workload spike, the user would like a VM to quickly migrate on-demand to faster and more expensive cores – which is possible with heterogeneous-ISA machines, but requires an hypervisor.

Finally, virtualizing heterogeneous-ISA hardware enables the classical features of virtualization: heterogeneous VMs will benefit

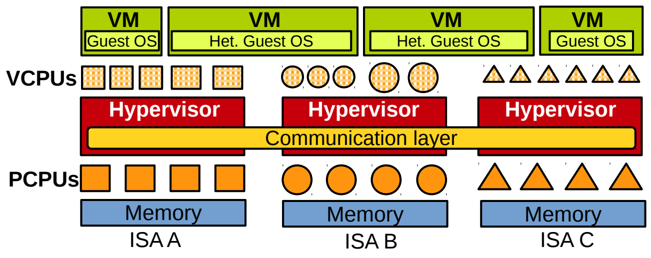

Figure 1: The multihype design, in which multiple hypervisors instances, each managing a set of physical CPUs of the same ISA, abstracts a heterogeneous ISA machine hardware to create multiple, potentially heterogeneous, VMs.

from consolidation, checkpoint/restart, migration, sandboxing, VM introspection, etc.

Heterogeneity and Multiple-kernel OSes. So far, the problem of executing applications among heterogeneous ISA processors has been considered by the OS community [1, 2, 4, 27, 32] that came up with the multiple-kernel OS design. A multiple-kernel OS instantiates different OS kernels on different cores, or group of homogeneous cores. In a heterogeneous processing platform, to support multiple ISAs, the address space of each kernel instance is different – e.g., at minimum the code section of each instance includes machine instructions for a specific ISA. The multikernel [4] is the most popular multiple-kernel design, and it has influenced a plethora of OS works [1, 2, 27, 32]. The multikernel is based on the idea that kernel instances do not use shared memory. Instances are "shared nothing", therefore they don't share any part of their address space and communicate only via message-passing. For this reason, OS services, which may be spawned on multiple kernel instances, are implemented as distributed services – based on distributed protocols.

3 The Multihype Design & Architecture

Scope and Assumptions. In our design, CPU virtualization is provided through direct-execution [8]. Existing solutions [5, 6] based on emulation suffer from slowdowns that are unacceptable in production. Thus, we assume hardware (full) virtualization support for all processing units composing a heterogeneous platform. Anyhow, hardware virtualization technologies are available in most modern processors, including emerging ISAs, such as RISC-V [7].

Note that while paravirtualization can help with non-virtualizable architectures [8], it drifts away from the pure virtualization concept of providing the same machine interface to the software running atop an hypervisor, by extending such machine interface with additional APIs – the hypercalls. Therefore, a paravirtualized hypervisor maybe though such as a microkernel [19], thus it can be extended to handle heterogeneity by means of the multikernel design.

The Multihype. We define a multihype as a hypervisor built with different hypervisor instances. Each instance is compiled for a given ISA/microarchitecture and runs on a core or set of cores of the same ISA, ISA extensions, and microarchitecture – i.e., a processor island. These instances communicate with each other to maintain a consistent state for each running VM. VMs are virtual versions of the physical host. This involves creating heterogeneous virtual CPUs as well as different pseudo-physical address spaces from the physical CPUs and memory on the host. A fundamental feature here is that the multihype can create a VM with heterogeneous VCPUs. An illustration of this design is presented in Figure 1. Note that in theory the multihype can also apply emulation (Dynamic Binary Translation) to create a homogeneous-ISA VM on top of a heterogeneous host, although the benefits of such a setup are unclear. Our design does not make assumptions about how processors running different hypervisor's instances are interconnected – it can be anything, including Ethernet, PCIe, coherency bus, etc. Therefore, the multihype does not define a sharing model between hypervisor instances, nor a communication model, such as message-passing or shared memory. Moreover, our design is not specific to type 1 or 2 hypervisors, and instances can either run bare metal or be hosted in a modified host OS such as a multikernel.

The main features of a multihype design are: 1) inter-hypervisor communication; 2) presenting the same machine interface than the virtualized physical machine, including virtual CPUs, memory, devices, and eventual features emulation, including CPU instruction extensions, DSM with different consistency properties when share memory is not present, and virtual devices; and 3) providing booting, termination, memory, scheduling, and attestation services seamlessly across hypervisors. A multihype can run VMs composed of both homogeneous- and heterogeneous-ISA processors. In the latter case, we obviously assume the use of a guest OS supporting heterogeneity, such as potentially adapted versions of Barrelfish [4], Popcorn [2].

Communication between hypervisor instances is fundamental in order to share resources and coordinate to manage VMs. When shared memory is available, it is preferred for performance reasons. Without shared memory, message passing is the paradigm of choice. Although in our experience [1, 2] multihype should be built from the same portable code-base, this is not a strict requirement – but it helps in fixing a communication format.

A multihype may provide support for heterogeneous ISA processors in terms of emulating specific functionalities of instructions. This may also include hypervisor services such as distributed shared memory for hypervisor instances that cannot rely on shared memory interconnects.

Finally, a multihype runs a minimal set of coordinated services on each hypervisor instance: memory management, VCPU scheduling, and interrupt management. Such services are written in order to exploit messaging and/or shared memory to communicate between different instances[1]. All other services (VM creation/deletion, migration, paravirtualized drivers, etc.) are coordinated and implemented either by one or a set of privileged VMs [11], such as Xen's domain 0 for type 1 hypervisor deployments, or by the host for type 2 systems.

Virtual CPU Migration. A multihype will be able to migrate a set of heterogeneous VCPUs from one heterogeneous physical machine to another. However, we believe a multihype has further potential on which we can tap in. VCPU/VM migration across compute units of heterogeneous ISAs would be highly beneficial in some scenarios, for example when elasticity is needed. However, it is challenging because it would require heterogeneity support from the guest OS itself. While it is possible for processes to perform such

[1]The authors have another submission pending which explains how multiple-kernel or multiple-hypervisor may communicate on shared memory

migration as demonstrated by the Popcorn Linux OS and Compiler Framework [1], adopting the same mechanism for the kernel is not possible due to the architecture-specific characteristics of the kernel state.

We rather propose to solve this migration issue by extracting the essential part of a VM execution, i.e. the application state, and re-initializing a fresh kernel for the target ISA. Translation to the target ISA can be applied to the architecture-specific application state, for example in a similar manner as defined in Popcorn Linux [1], then restored in the freshly booted kernel. Obviously, instead of a freshly booted kernel, a pre-booted image (VM fork [25]) can be resumed in order to speed up the described migration process. Note that it is likely that this process will require substantial communication between the guest and hypervisor to bridge the semantic gap, and we do not exclude the development of dedicated hypercalls/upcalls. We believe that for infrequent migrating VM this approach is superior to ISA emulation.

Implementation Sketch. We are building a prototype implementing the multihype model within the popular KVM hypervisor [24]. There is currently no real-world SoC with heterogeneous-ISA CPUs. Existing work either rely on simulation [13, 40] or FPGA [26] prototypes; and we were still not able to put our hands on the Intel's Xeon Gold 6138P processor. Our implementation will then execute on a custom setup composed of two separate machines: a x86-64 (AMD Epyc 7451) and an ARM server (Cavium ThunderX). The servers are interconnected via Dolphin PXH810 PCIe 3 8x adapters [39] providing sub-microsecond shared memory as well as DMA transfers. This setup is just a scaled up version of any x86 machine with an intelligent SSD or NIC [16, 29–31] plugged in.

Because such a setup is composed of physically separated machines, our implementation of multihype may resemble a distributed hypervisor. Communication between hypervisor instances are leveraged to distribute memory (through a DSM implementation), VCPUs of different ISAs, interrupts, as well as devices. As a result, a VM has the illusion of running on a shared-memory heterogeneous-ISA machine. Such a virtual model, virtualizing CPUs through direct-execution, probably present better performance than any existing prototype of future shared-memory heterogeneous-ISA platforms such as software models [13, 40] or FPGA implementations [26]. This model can be used for research concerning heterogeneous OS on such platforms which real-world implementations are not available yet.

As a first step we plan to develop an implementation capable of consolidating multiple heterogeneous VMs from a physical heterogeneous setup. We propose to adapt and run Popcorn Linux [1] as the guest OS on these VMs. As a second step, we will focus on VCPU migration among heterogeneous processing units. Application state and the non architecture-specific part of the kernel state related to an application (such as file descriptors) can be extracted with Checkpoint-Restart In Userspace [12] (CRIU). If using Linux for heterogeneous VCPU migration ends up being too challenging, we plan to fall back on simpler guest OSes, namely unikernels [28].

4 Conclusion

Computer architectures are increasingly populated by heterogeneous ISA general-purpose processors. At the same time, virtualization technologies are being adopted more broadly – not just in the data center. Both heterogeneity and virtualization are taking over different markets from server to embedded. Thus, we considered the problem of how a VM for a set of heterogeneous processors should look like and be built. We propose the multihype design together with an initial sketch of its implementation. The multihype applies ideas from the multiple-kernel OS design into hypervisor design.

Acknowledgments

This work is supported in part by ONR under grants N00014-16-1-2104 and N00014-16-1-2711.

References

[1] Antonio Barbalace, Rob Lyerly, Christopher Jelesnianski, Anthony Carno, Ho-ren Chuang, and Binoy Ravindran. 2017. Breaking the Boundaries in Heterogeneous-ISA Datacenters. In *Proceedings of the 22th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17)*.

[2] Antonio Barbalace, Marina Sadini, Saif Ansary, Christopher Jelesnianski, Akshay Ravichandran, Cagil Kendir, Alastair Murray, and Binoy Ravindran. 2015. Popcorn: Bridging the Programmability Gap in heterogeneous-ISA Platforms. In *Proceedings of the Tenth European Conference on Computer Systems (EuroSys '15)*. 29:1–29:16.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the art of virtualization. In *ACM SIGOPS operating systems review*, Vol. 37. ACM, 164–177.

[4] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania. 2009. The Multikernel: A New OS Architecture for Scalable Multicore Systems. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles (SOSP '09)*. 29–44.

[5] Fabrice Bellard. 2005. QEMU, a fast and portable dynamic translator.. In *USENIX Annual Technical Conference, FREENIX Track*, Vol. 41. 46.

[6] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.

[7] Paolo Bonzini, John Hauser, and Andrew Waterman. 2017. RISC-V Hypervisor Extension. 7th RISC-V Workshop.

[8] Edouard Bugnion, Jason Nieh, and Dan Tsafrir. 2017. Hardware and software support for virtualization. *Synthesis Lectures on Computer Architecture* 12, 1 (2017), 1–206.

[9] George Chrysos. 2014. Intel® Xeon PhiâĎć coprocessor-the architecture. *Intel Whitepaper* 176 (2014).

[10] Hongsuk Chung, Munsik Kang, and Hyun-Duk Cho. 2012. Heterogeneous Multi-Processing Solution of Exynos 5 Octa with ARM® big. LITTLEâĎć Technology. *Samsung White Paper* (2012).

[11] Patrick Colp, Mihir Nanavati, Jun Zhu, William Aiello, George Coker, Tim Deegan, Peter Loscocco, and Andrew Warfield. 2011. Breaking up is hard to do: security and functionality in a commodity hypervisor. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 189–202.

[12] CRIU Contributors. 2019. CRIU Website. https://criu.org/Main_Page, Online, accessed 2019/01/10.

[13] Matthew DeVuyst, Ashish Venkat, and Dean M. Tullsen. 2012. Execution Migration in a heterogeneous-ISA Chip Multiprocessor. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII)*. 261–272.

[14] José Duato, Antonio J Pena, Federico Silla, Rafael Mayo, and Enrique S Quintana-Ortí. 2010. rCUDA: Reducing the number of GPU-based accelerators in high performance clusters. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*. IEEE, 224–231.

[15] Peng Fan. 2016. Question About KVM on Big.Little SoC. https://www.spinics.net/lists/kvm/msg137631.html, Online, accessed 01/15/2019.

[16] Boncheol Gu, Andre S Yoon, Duck-Ho Bae, Insoon Jo, Jinyoung Lee, Jonghyun Yoon, Jeong-Uk Kang, Moonsang Kwon, Chanho Yoon, Sangyeun Cho, et al. 2016. Biscuit: A framework for near-data processing of big data workloads. In *ACM SIGARCH Computer Architecture News*, Vol. 44. IEEE Press, 153–165.

[17] Prabhat K Gupta. 2015. Xeon + FPGA Platform for the Data Center. In *Fourth Workshop on the Intersections of Computer Architecture and Reconfigurable Logic*, Vol. 119.

[18] Vishakha Gupta, Ada Gavrilovska, Karsten Schwan, Harshvardhan Kharche, Niraj Tolia, Vanish Talwar, and Parthasarathy Ranganathan. 2009. GViM: GPU-accelerated virtual machines. In *Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing*. ACM, 17–24.

[19] Steven Hand, Andrew Warfield, Keir Fraser, Evangelos Kotsovinos, and Daniel J Magenheimer. 2005. Are virtual machine monitors microkernels done right?. In *HotOS*.

[20] Yu-Ju Huang, Hsuan-Heng Wu, Yeh-Ching Chung, and Wei-Chung Hsu. 2016. Building a kvm-based hypervisor for a heterogeneous system architecture compliant system. In *ACM SIGPLAN Notices*, Vol. 51. ACM, 3–15.

[21] Intel. 2018. Fact Sheet: New Intel Architectures and Technologies Target Expanded Market Opportunities. https://intel.ly/2FFfwig, Online, accessed 01/10/2019.

[22] Intel. 2018. Intel Xeon Processor Scalable Family. https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/xeon-scalable-datasheet-vol-1.pdf, Online, accessed 01/10/2019.

[23] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*. IEEE, 1–12.

[24] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. 2007. kvm: the Linux virtual machine monitor. In *Proceedings of the Linux symposium*, Vol. 1. Dttawa, Dntorio, Canada, 225–230.

[25] Horacio Andrés Lagar-Cavilla, Joseph Andrew Whitney, Adin Matthew Scannell, Philip Patchin, Stephen M Rumble, Eyal De Lara, Michael Brudno, and Mahadev Satyanarayanan. 2009. SnowFlock: rapid virtual machine cloning for cloud computing. In *Proceedings of the 4th ACM European conference on Computer systems*. ACM, 1–12.

[26] Katie Lim, Jonathan Balkind, and David Wentzlaff. 2018. Juxtapiton: Enabling heterogeneous-isa research with RISC-V and SPARC FPGA soft-cores. *arXiv preprint arXiv:1811.08091* (2018).

[27] Felix Xiaozhu Lin, Zhen Wang, and Lin Zhong. 2014. K2: A Mobile Operating System for Heterogeneous Coherence Domains. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. 285–300.

[28] Anil Madhavapeddy and David J Scott. 2014. Unikernels: the rise of the virtual library operating system. *Commun. ACM* 57, 1 (2014), 61–69.

[29] Marvell. 2019. LiquidIO II 10/25GbE Adapter family. https://www.marvell.com/ethernet-adapters-and-controllers/liquidio-smart-nics/liquidio-ii-smart-nics/, Online, accessed 2019/02/11.

[30] Timothy P. Morgan. 2013. Tilera Rescues CPU Cycles with Network Coprocessors. https://bit.ly/2DfM53R, Online, accessed 01/05/2019.

[31] Netronome. 2019. About Agilio SmartNICs. https://www.netronome.com/products/smartnic/overview/, Online, accessed 01/05/2019.

[32] Edmund B Nightingale, Orion Hodson, Ross McIlroy, Chris Hawblitzel, and Galen Hunt. 2009. Helios: heterogeneous multiprocessing with satellite kernels. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 221–234.

[33] Vlad Nitu, Pierre Olivier, Alain Tchana, Daniel Chiba, Antonio Barbalace, Daniel Hagimont, and Binoy Ravindran. 2017. Swift birth and quick death: Enabling fast parallel guest boot and destruction in the xen hypervisor. In *ACM SIGPLAN Notices*, Vol. 52. ACM, 1–14.

[34] NVIDIA. 2014. NVIDIA Virtual GPU Technology. https://www.nvidia.com/en-us/design-visualization/technologies/virtual-gpu/, Online, accessed 01/15/2019.

[35] Pierre Olivier, Sang-Hoon Kim, and Binoy Ravindran. 2017. OS Support for Thread Migration and Distribution in the Fully Heterogeneous Datacenter. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. ACM, 174–179.

[36] Vidya Rajagopalan, Vamsi Boppana, Sandeep Dutta, Brad Taylor, and Ralph Wittig. 2011. Xilinx Zynq-7000 EPP: An extensible processing platform family. In *Hot Chips 23 Symposium (HCS), 2011 IEEE*. IEEE, 1–24.

[37] Phil Rogers. 2013. Heterogeneous system architecture overview. In *Hot Chips*, Vol. 25.

[38] Lin Shi, Hao Chen, Jianhua Sun, and Kenli Li. 2012. vCUDA: GPU-accelerated high-performance computing in virtual machines. *IEEE Trans. Comput.* 61, 6 (2012), 804–816.

[39] Dolphin Interconnect Solutions. 2015. PXH810 PCI Express Gen3 Host Adapter. https://www.dolphinics.com/download/PX/OPEN_DOC/PXH810_Product_Brief.pdf, Online, accessed 01/10/2019.

[40] Ashish Venkat and Dean M. Tullsen. 2014. Harnessing ISA Diversity: Design of a heterogeneous-ISA Chip Multiprocessor. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture (ISCA '14)*. IEEE Press, Piscataway, NJ, USA, 121–132. http://dl.acm.org/citation.cfm?id=2665671.2665692

[41] Xen Contributors. 2018. Xen Documentation - ARM big.LITTLE Support. https://xenbits.xen.org/docs/unstable/misc/arm/big.LITTLE.txt, Online, accessed 01/15/2019.