

RH RQ

Bringing great research ideas
into open source communities

Václav Matyáš

open source cybersecurity
and the next generation

Machine learning
meets big data

Finding bugs in
parallel programs

+

Mental models



A thread model:

Daniel Bristot de
Oliveira on the
formal analysis
and verification
of the real-time
Linux kernel



Red Hat
Research Quarterly
Volume 2:3 | October 2020 | ISSN 2691-5278



DID YOU KNOW?

SAS[®] BRINGS ARTIFICIAL INTELLIGENCE AND ANALYTICS TO THE CLOUD.

You can run SAS on private, public or hybrid cloud infrastructures to better manage how AI work is done. SAS works with all major cloud providers to give you the power and freedom to innovate and be agile in the cloud.

sas.com/discover



Table of Contents



Departments

- 04** From the director
- 05** Column: Better together
- 06** News: Research at Devconf.us
- 34** Research project updates

Features

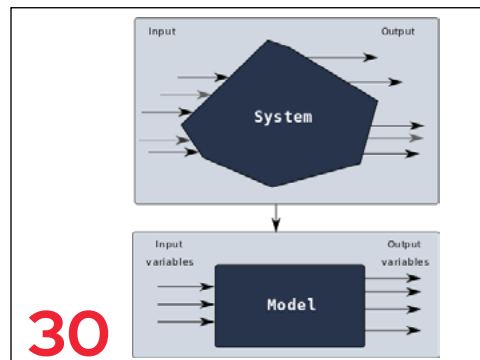
- 08** Machine learning meets big data
- 11** Finding bugs in parallel programs
- 16** Mental models
- 20** Open source cybersecurity and the next generation: an interview with Václav Matyáš
- 30** A thread model for the real-time Linux kernel

THE BUILDER **THE MANAGER**

THE GENERALIST



16



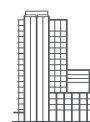
facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

NORTH AMERICA
1 888 REDHAT1

EUROPE, MIDDLE EAST,
AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.com



ABOUT RED HAT Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

From the Director

Untangling complex systems

by *Hugh Brock*


It is by now well understood that we humans are capable of creating systems that are more complex than we can understand. I would venture to go a bit further with this and say that many of us like creating complex systems, the more complex the better, usually in the service of trying to be both useful and elegant at the same time. Nowhere is this truer than in software development, where the toll of complexity need not be paid until months or even years after its creation, and even then can be deferred by wrapping an incomprehensible “black box” with another layer that we believe we understand. This happens all too frequently, and the end result can be systems so complex that they can only be understood by other systems.

It so happens that this issue of RHRQ has several articles that touch on different aspects of this problem, beginning with our interview with security researcher Václav Matyáš. Professor Matyáš, who specializes in computer security and, lately, in the problem of usable security, has been working with Red Hat for a very long time; he might, in fact, be the very beginning of Red Hat Research itself. Red Hat security architect Mike Bursell spoke with him at length about the problem of making computer security—an inherently complex topic—simple enough to work for real people, among many other interesting topics.

Other examinations of how to approach complexity come in two articles on code analysis and formal verification. Vladimír Štill examines the long-running Brno research project DIVINE, the subject of his and many others’ PhD work. The project provides a new approach to the very complex topic of debugging multi-threaded programs. If the words “race condition” fill you with cold dread, you will want to read this piece. Meanwhile, Daniel Bristot de Oliveira reviews his own PhD work on modelling the Linux kernel using finite automata. Breaking complexity into chunks of observable behavior can make it much easier to deal with, it turns out.

Our final piece on complexity, from researcher and Red Hatter Ilya Kolchinsky, surveys different possible techniques for using neural networks to solve machine-native problems—problems that are inherently too large or complex to be solvable by a human. There are many emerging possibilities here.

Reading articles like these gives me some hope that we may be able to develop tools that will help us untangle some of the complexity we create. At the same time, we will of course learn ways to build even more complex systems, aided by these very same tools.

Perhaps some day we will build tools that actually make things simpler? I’ll believe it when I see it... 

**About the Author**

Hugh Brock is the Research Director for Red Hat, coordinating Red Hat research and collaboration with universities, governments, and industry worldwide. A Red Hatter since 2002, Hugh brings intimate knowledge of the complex relationship between upstream projects and shippable products to the task of finding research to bring into the open source world.



Better together

Thoughts on open source and open collaboration from the Greater Boston Research Interest Group (RIG)

by Beverly Kodhek

There's been a lot of emphasis placed on collaboration in the workplace. And while it's easy to add collaboration to the ever-growing list of important company-wide objectives, implementing it requires a more structured plan and a clear understanding of current employee workflows. When leaders try to establish the reasons why collaboration has not taken hold, more often than not they find that it relates to organization structure and how information sharing is prioritized.


Red Hat is an open source company, and our decentralized culture encourages open collaboration, not only internally, but with upstream communities too. It is this open culture that has contributed to our success as an organization. Within the research team, we encourage collaboration through Research Interest Groups (RIGs). The goal of each RIG is to foster research that aligns with Red Hat's technical direction and open source mission. Through the RIGs, researchers at our partner universities work collaboratively with Red Hat teams and seek opportunities for

conducting innovative research on specific technology topics. Some of the topics we are currently exploring together include operating systems, machine learning and automation, and cloud computing services.

Nonetheless, RIGs are not just about adding more people. There is certainly an art to this collaboration. The Boston RIG, for example, has attracted participation from professors, engineers, PhD candidates, interns, marketers, and patent lawyers. This integration of varied perspectives creates that sweet spot for taking conventional ideas and applying them to interesting problems in a novel way to produce groundbreaking solutions.

The Greater Boston RIG has also been able to take advantage of partnerships with universities in the area to tap into an excellent talent pool of potential hires for engineering and other technology roles. During the course of this summer, we worked with 140 interns who all got an opportunity to share their project ideas in the form of lightning talks. These ten-minute sessions were

meant to share the essence of an idea and get more managers and engineers involved in the projects. Not only that, they were also a great opportunity to build a community around projects.

So if you are looking to create the next breakthrough product, teaming up a group of specialists in novel ways might be a good idea. Research Interest Groups can help with that. Even though we are not able to meet in person now, we can still make space for people to meet potential collaborators they may not otherwise run into. I urge you to organize a Research Interest Group and foster collaboration within your organization. 

About the Author Beverly Kodhek

is part of the product marketing team at Red Hat Research. She develops messaging and implements strategic initiatives to grow Red Hat Research both internally at Red Hat and within the broader ecosystem of partners.



Research at Devconf.us: Optimizing and automating the foundations of computing

Software foundations like operating systems and hypervisors—to say nothing of the server hardware itself—are boring. Or at least that’s how almost everyone working atop them wants them to be. Who wants an exciting foundation when you’re trying to get your job done?

But there’s still plenty of innovative engineering work going on in those low layers, and cutting-edge research too. The latter was highlighted at Devconf.us in talks by PhD students at Boston University including Ali Raza and Tommy Unger, Han Dong, and Parul Sohal.

LOW-LEVEL HARDWARE AND SOFTWARE OPTIMIZATIONS DON’T TAKE ONE FORM

Ali Raza’s research (research.redhat.com/blog/research_project/unikernel-linux/) focuses on unikernels. The idea of the unikernel is that you build your app with the kernel it’s going to run on so that you’ve basically built a bootable app. Advantages include fast booting, a reduced attack surface, and a shorter path from the app to system calls. The current Linux kernel in this research hasn’t been slimmed down to just the basics (library kernel) yet, but a great deal of progress has been made.

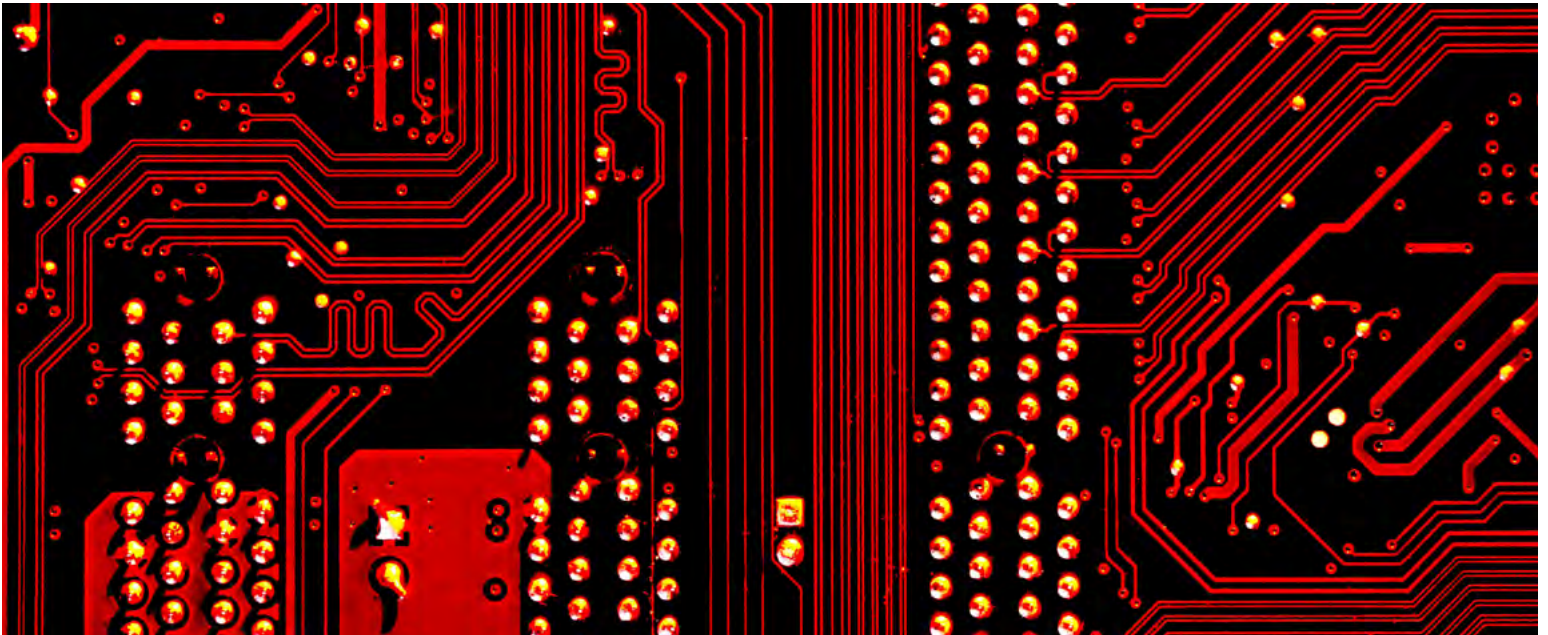
Raza’s co-presenter, Tommy Unger, is working on the hypervisor layer. Like unikernels, hypervisors can be smaller and therefore offer a potentially smaller attack surface than a full-blown operating system kernel. Nonetheless, because they are both ubiquitous and essential, they are security-critical applications that make attractive targets for potential attackers. Virtual devices are a common site for security bugs in hypervisors. Unger’s work has focused on a novel way of fuzzing virtual devices (an automated software testing technique) that combines a standard

coverage-guided strategy with further guidance based on hypervisor-specific behaviors.

Parul Sohal’s research interests lie in the management of resources at different levels of the memory hierarchy (Quality Of Service included). Her goal is to achieve better resource utilization and isolation so as to avoid contention, which

causes application performance degradation below a minimum quality of service. Sohal’s work takes advantage of recent Intel processor features such as reserving a subset of cache for a given program and memory bandwidth throttling. Combined with containers and control groups (Cgroups), these features can help prevent programs from interfering with each other, something often called the noisy neighbor problem.

The current Linux kernel in this research hasn’t been slimmed down to just the basics (library kernel) yet, but a great deal of progress has been made.



Han Dong's work highlights some of the challenges of tuning software and hardware. Dong observes that a modern network interface card (NIC), such as the Intel X520 10 GbE, is complex, with thousands of hardware registers that control every aspect of the NIC's operation, from device initialization to dynamic runtime configuration. That's far too many tuning parameters for a human to manually configure, and only about a third of them are even initialized by today's Linux kernel. The goal of Dong's research is to automate tuning this NIC using machine learning.


WHAT HAPPENS AT THE SYSTEM LEVEL?

However, if we now step back and take a look at the bigger performance and optimization picture, a challenge emerges. While specific optimizations

often happen at a very detailed micro-level—as in the case of the operating system, hypervisor, processor cache, or NIC—the real goal is to optimize at the system (or even the datacenter) level. And just as individual programs can suboptimally compete for resources on a single processor, so too can individual low-level optimizations lead to undesirable side effects at the global system level.

As Red Hat Senior Distinguished Engineer Larry Woodman puts it, "Several new CPU/hardware features whose implementation is not yet well understood are likely to conflict with each other when running different applications and benchmarks, causing nondeterministic performance behavior.

"Understanding these patterns given so many variables soon becomes a

daunting task for anyone. For this reason it's likely that Red Hat Research will investigate automating this process by deploying artificial intelligence / machine learning (AI/ML) techniques and algorithms to uncover and attempt to fix a wide range of scenarios. A future project Red Hat Research is investigating involves using AI/ML for overall system configuration and, ultimately, automated tuning. There are so many parameters that adversely affect each other that manual or even profile-based tuning is not effective or even possible." 

You can follow Red Hat Research projects, and even suggest a project based on open source software, at research.redhat.com. Recordings of Devconf presentations are available at www.youtube.com/c/DevConf_INFO/playlists.

Feature

When machine learning meets big data processing: From human-native tasks to machine-native tasks

Since the inception of artificial intelligence research, computer scientists have aimed to devise machines that think and learn like human beings. What else could AI do?



by *Ilya Kolchinsky*

Image classification, language-to-language translation, and speech recognition are some of the most prominent examples of major tasks attributed to humans in which great success has been achieved by modern machine learning technologies.

Unfortunately, as a consequence of this vision, important tasks that are not perceived as *human native* are commonly neglected by most AI-related research communities. This category of problems, which we call *machine native*, is characterized by: 1) being unsolvable by a human without the aid of a computer, and 2) the existence of a known, not necessarily efficient algorithm capable of solving the problem. Hard combinatorial optimization problems such as the traveling salesman problem or finding the maximum clique in a graph are obvious examples of this category. Many practical machine-native tasks have virtually no known efficient solutions and could benefit greatly from approaches based on the recent groundbreaking achievements in machine learning.

In this article, we will provide a glimpse into a number of ongoing research directions addressing this second type of AI-assisted tasks in the context of the category of computer systems collectively known as *big data processing systems*.

BIG DATA (STREAM) PROCESSING

As we enter the era of big data, a large number of data-driven systems and applications have become an integral part of our daily lives, and this trend is accelerating dramatically. It is estimated that 1.7MB of data are created every second for every person on earth, for a total of over 2.5 quintillion bytes of new data every day, reaching 163 zettabytes by 2025 according to the International Data Corporation. Many practical challenges encountered by modern big data systems are further exacerbated by the growing volume, velocity, and variety of continuously generated data, presented to them in the form of near-infinite *data streams*. The complexity of big data processing systems grows over time, together with user requirements and data volume, and increases exponentially with system scale.

About the Author

Ilya Kolchinsky is a research scientist with Red Hat Research and Technion, Israel Institute of Technology. He has a PhD and BSc in Computer Science, both from the Technion. Ilya's research interests span a wide range of topics in big data processing, such as distributed event-based systems, data stream mining, and applications of AI and machine learning in stream processing engines.

A typical big data processing application involves hundreds to thousands of *operators* connected by communication channels to form a directed graph referred to as a *data processing network*. This network is constructed according to the dedicated *query evaluation plan*, which is derived from the queries submitted by the system users. For the most part, each operator is relatively simple and serves a generic purpose, whereas their composition in every segment of the data processing network implements an application-specific requirement.

BIG DATA PROCESSING OPTIMIZATION USING DEEP REINFORCEMENT LEARNING

The problem of data processing optimization dates back to the inception of early database systems. The input to this problem is a user query scheduled for execution, and the task is to convert this query into a series of low-level operations comprising an evaluation plan. The same query could correspond to multiple possible evaluation plans. For example, if a user wishes to extract and combine data from n tables, there are $n!$ orders of accessing these tables. Even more possibilities are introduced if the target system contains multiple implementation options for some of the operators, if the computation can be distributed over multiple nodes, etc. As different plans could have differences of a few orders of magnitude in their performance characteristics, such as execution time and resource consumption, the task of selecting the optimal plan is of utmost importance for any data processing system.

Picking the best performing evaluation plan is a challenging task due to the extremely high number of possible solutions. Since the early 70s, a plethora of methods and algorithms has been developed to attempt to solve this problem. In spite of these efforts, existing solutions often prove either inefficient or imprecise. A plan optimization algorithm cannot afford

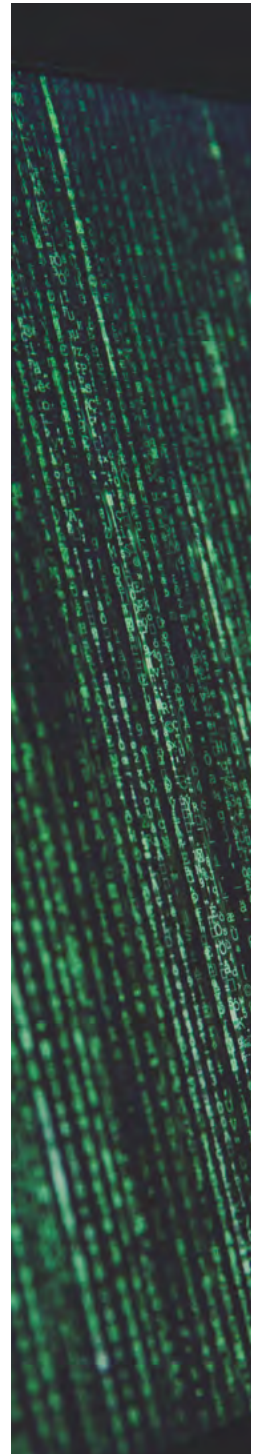
to scan the huge plan space or a substantial fraction thereof and is instead forced to utilize heuristics, which might or might not work.

One approach that could come to the rescue here is known as *deep reinforcement learning*—the very same method that gained fame as the driving force behind AI-based chess, backgammon, and Go players. In reinforcement learning, the trained model learns to perform sequences of moves leading to states providing maximum reward, such as a victory in a game. The learning process is performed by way of trial and error, and *deep neural networks* are utilized to handle the huge possible state space. In the data processing optimization domain, the process of crafting an efficient query evaluation plan could be considered a “game,” with a set of “moves” defined as all possible selections and placements of an operator in a particular position. By continuously creating plans, applying them on sample data, and measuring the resulting performance, an optimizer implementing this paradigm could gradually learn the most efficient plans.

DEEP NEURAL NETWORKS AS AN EFFICIENT ALTERNATIVE TO TRADITIONAL BIG DATA PROCESSING MECHANISMS

One could suggest an alternative approach to the query processing optimization problem discussed above. Instead of devising smart algorithms for arranging the operators into an efficient evaluation plan, why not take a step further and replace the entire data processing engine with a pretrained deep neural network capable of answering the query?

While seemingly unrealistic at first, this idea has a number of clear advantages. First, since a neural network merely approximates the expensive computation that a query processing engine directly performs, the former is expected to run



considerably faster and to consume fewer resources. For example, if the user-defined query is to correlate between two data streams A and B and to find all pairs of A's and B's satisfying a predefined condition, the neural network will not have to actually compare between all candidate A-B pairs but instead will settle for a cheaper computation based on the function it learned during training. Second, since the inference time (i.e., the time required to provide an output given an input) of a trained network is constant, the need for using complex optimization methods and algorithms for maximizing the performance of a query evaluation plan would become obsolete.

The main disadvantage of the neural network-based data processing approach is the possibility of returning imprecise or erroneous results due to the imperfection of the learning process. An ongoing challenge for research is to find ways to achieve high levels of precision by utilizing ensemble methods or other novel regularization techniques. In addition, trading off result accuracy (up to a certain level) for performance is acceptable or even highly desirable in many modern big data applications.

PREDICTING FUTURE BIG DATA STREAM QUERY RESULTS

As indicated above, the primary task of a big data engine is to deliver up-to-date query results to the end users. It might be even more useful to go a few steps forward and predict the future returned values based on the observed trends in the continuously generated streaming data. Such functionality

could be highly beneficial in real-time processing scenarios where a particular action must be triggered and promptly executed immediately (typically within milliseconds) following an occurrence of a particular data item or a combination thereof, and where even the most prolific data processing techniques fail to provide a sufficiently small detection latency. Furthermore, in some situations the goal is to prevent a certain event from occurring rather than react to it, a use case that cannot be realized without an ability to predict the future state with some degree of confidence.

...a future generation of big data processing engines could offer a new capability of accurately predicting query answers that will enhance the proactive response abilities of user applications.


For a data processing system to provide future query answers, there is a need to get a snapshot of the expected future data values. The long-established field of *time series forecasting* was designed to do exactly that. An increasingly active area of research, it received an unprecedented boost in recent years following a breakthrough in deep learning. It was demonstrated by multiple research teams around the world that certain types of neural networks (such as LSTM, TCN, and Transformer) could achieve

remarkable success in learning the data trends and predicting the future data stream content based on past history. While many data analytics and stream analytics frameworks provide time series forecasting as a separate feature, typically as a part of a larger data mining package, incorporating this technology into the core of the query processing engine is yet to become a major trend.

By combining a state-of-the-art time series forecasting method and an efficient mechanism for processing the raw data and acquiring the query results (such as one of those described above), a future generation of big data processing engines could offer a new capability of accurately predicting query answers that will enhance the proactive response abilities of user applications.

WHAT DOES THE FUTURE HOLD?

In this short article, we have barely scratched the surface of the immense unrealized potential of machine learning in the area of big data processing. In the Technion University research team, undergraduate and graduate students are working side by side to produce innovative solutions for these and many other open challenges for practical problems in human-native, machine-native, and hybrid problem domains.

We are looking for projects that will help us test these techniques. Those interested in finding out more about our project ideas and/or looking for collaboration opportunities are kindly invited to contact Dr. Ilya Kolchinsky at ikolchin@redhat.com. 

Finding bugs in parallel programs with heavy-duty program analysis

Parallelism promises to make programs faster, yet it also opens many new pitfalls and makes testing programs much harder.

By *Vladimír Štill*

Anyone who has ever tried to write a piece of parallel software knows it is not an easy task. Parallelism promises to make programs faster, yet it also opens many new pitfalls and makes testing programs much harder. For example, writing and reading a variable in parallel from two threads may seem to work, but it will backfire sooner or later unless the variable is somehow protected by a lock or using some language primitives to make the accesses atomic.

The worst part about these problems is that they often manifest themselves only under specific timing of the involved threads or on a particular platform. Therefore, a program might run smoothly most of the time, but might fail once every few minutes or even once every few months. A service failing once every few months on a customer's server is surely among developers' worst nightmares. Knowing this problem exists, can we help developers find problems in parallel software?

We believe heavy-duty program analysis is one option that can help.

THE PROBLEM

To meaningfully use parallelism, threads of a parallel program often need to communicate with each other. In this case, it is the responsibility

of programmers to ensure that all of the communicating threads have a consistent view of the memory and therefore can work correctly. For example, consider an insertion to a doubly linked list. The thread that performs the insertion will create a new node, set its value, and set the pointers to the previous and next node inside the node-to-be-added. So far no problem can happen in parallel execution: the new node is not yet visible to the other threads, and it is not linked into the list.

However, consider that at this point another thread performs insertion to the same place in the linked list. Now the list has changed, but the pointers in the node-to-be-added do not reflect this change. If the node is inserted anyway, we will get an inconsistent list. We might miss some of the inserted elements when iterating over it, or even find different elements when iterating forward and when iterating backward. The root cause of this problem is that the addition of a node to the linked list is not an atomic operation. That is, it can be completed partially, and other threads might be affected by this partial result.

There is no single solution to our example problem. It might be possible to avoid accessing the list from multiple threads altogether. It might also be



About the Author

Vladimír Štill is a PhD candidate at the Faculty of Informatics, Masaryk University in Brno. His research topic is the correctness analysis of parallel programs written in C and C++. His work includes finding new analysis techniques and their implementation in the DIVINE tool.

DIVINE is a modern, explicit-state model checker. Based on the LLVM toolchain, it can verify programs written in multiple real-world programming languages, including C and C++. The verification core is built on a foundation of high-performance algorithms and data structures, scaling all the way from a laptop to a high-end cluster. The name “LLVM” itself is not an acronym; it is the full name of the open source project (see llvm.org).

DIVINE is free software, distributed under the ISC licence (2-clause BSD). You can find more information about this project, download the software, or simply follow our progress at <https://divine.fi.muni.cz/index.html>.

sufficient to add a lock that protects accesses to the list or to make the list use locks internally. Such locks will prevent two threads from inserting at once into the same list. It might also be necessary to design the whole program in such a way that it can use some high-performance lock-free data structures for the communication. In all but the first case, we are entering the realm of parallel programming, and we need to consider all its implications and risks.

One of the significant difficulties with parallel programs is that they are hard to test. This problem is caused by their dependence on timing. For example, our linked list example might work just fine if it so happened that the insertions are never executed at the same time during our testing. However, this does not mean that they cannot be executed at the same time. For this reason, tests might not fail for a buggy program, or they might fail only sometimes, making debugging harder.

Therefore, there is a need for tools that can help test parallel programs.

HEAVY-DUTY PROGRAM ANALYSIS

Many techniques that can improve testing parallel programs have been introduced. They have a wide range of complexity: from relatively fast code-analysis techniques, similar to compiler warnings, to more involved techniques like various thread sanitizers that can detect unsynchronised access to the same memory region from multiple threads. At the far end of the spectrum, there are heavy-duty analyses that essentially explore all possible executions of the parallel program. While the first two categories are relatively simple to use by developers, heavy-duty tools are still mostly academic projects that come with specific limitations. However, we believe their promises should not be ignored, even if they are far from being silver bullets.

Heavy-duty tools promise to check that a program does not do anything wrong (e.g., trigger an assert, access freed memory) regardless of the timing of threads. They can also take into account more advanced behaviors of the system, for example, relaxed memory present in most contemporary processors. Automated heavy-duty tools do this using many different basic techniques, such as stateless model checking, explicit-state model checking, symbolic model checking, and symbolic execution. But in essence, they explore all the possibilities for the timing of threads that can lead to different outcomes. The main difference between these techniques is in their basic approach to the complexity of the analysis. Each of these techniques comes with their limitations, and it is important to keep in mind that they are trying to solve a problem that is probably not solvable for every case. There will always be programs for which tools will fail or require many more resources than available. There are also similar tools that can explore all behaviors of the program based on all its possible inputs, and tools that combine both capabilities.

THE DIVINE ANALYZER

DIVINE is one of the heavy-duty analysis tools that can be applied to parallel programs. It was developed in the Parallel and Distributed Systems Laboratory at the Faculty of Informatics at Masaryk University in Brno, Czech Republic. DIVINE specializes in analysis of programs written in C++ (and C) and can handle both sequential and parallel programs. It can detect various

bugs including assertion violations, access out of memory bounds or to freed memory, use of uninitialized memory, and memory leaks.

To run DIVINE, it needs a test, and it can only detect problems that can happen in some execution of the test—i.e., it can try all possible ways that timing of threads and input data can influence the run of the test. This enables an entirely new way of writing tests of parallel programs. For example, instead of having to exercise the data structure with millions of elements inserted from several threads in the hope of triggering an elusive bug, it is sufficient to try it only with a few elements from two or three threads. DIVINE’s ability to explore all possible outcomes will mean such a test is sufficient (provided it exercises all features of the data structure, such as triggering growth in a test of a thread-safe hashset). Indeed, due to the computational complexity of program analysis, it is desirable to write tests for DIVINE that are as small as possible.

We will now look at some interesting recent additions to DIVINE with regards to parallel programs.

RELAXED MEMORY [^RMM]

In the struggle to construct more and more powerful processors, processor designers sometimes make decisions that make programmers’ lives harder. One of them is the use of relaxed memory to speed up memory access. Processors use caches, out-of-order execution, and speculative execution to mask the latency of the main memory. On most processor architectures, the presence of these mechanisms is observable by parallel programs. Maybe you have heard about the Meltdown and Spectre security vulnerabilities? They are caused by the same mechanisms that result in relaxed memory. While Meltdown and Spectre affect the security

of programs and operating systems, relaxed memory affects only parallel programs, but can cause them to crash or produce incorrect results.

Relaxed memory manifests itself differently on different hardware platforms. For the sake of simplicity, we will consider the x86-64 processors manufactured by Intel and AMD. These processors power most modern laptops, desktops, and servers. Other processors, e.g. high-performance ARM, are often even more relaxed. On an x86-64 processor, when a program stores data to a certain memory location, the processor does not wait for the store to finish before it executes the next instruction. Instead, it saves the stored value and its address internally into a *store buffer* that holds it until the store is committed to the memory. If the same CPU core that saved it reads the given location, it will get the value from the store buffer. Therefore, single-threaded programs behave as expected. However, if the same location is accessed by another thread running on another core while the store is in the store buffer, the new value is not yet visible to the other thread.

This can lead to very unintuitive behaviour. For example, consider a very simple program with two threads T_1 and T_2 and two global variables x and y (initialized to 0). Thread T_1 performs two actions: it assigns 1 to x ($x \leftarrow 1$) and reads y (read y). Thread T_2 has the variables switched: it performs $y \leftarrow 1$ and reads x . The question is what happens if both reads can read 0?

If we tried simulating these threads, we would probably conclude this cannot happen. At least one of the assignments has to happen before both of the reads, and therefore at least one of the variables has to be 1. However, thanks to the store buffers, both reads can return 0 on x86-64. For example, we can first execute all actions of

\$T_1\$: $\$x \leftarrow 1\$$; read $\$y\$$, clearly, the read returns 0. However, on x86-64, it is now possible the value 1 for variable $\$x\$$ is not yet saved in the memory but instead is in the store buffer corresponding to $\$T_1\$$. Therefore the memory still contains value 0 for $\$x\$$. Now we execute $\$T_2\$$: $\$y \leftarrow 1\$$; read $\$x\$$. Unless the store $\$x \leftarrow 1\$$ was already propagated to the memory, the read would return 0: a value which seem to be impossible from the inspection of the program.

Figure 1 (below) shows the actions against the store buffers as described above.

Programs have to use some sort of synchronization to prevent relaxed behaviour from breaking their programs. One option is to use locks: locks not only prevent two parts of the program that use the same lock from running at the same time, but they also ensure that all modifications performed before a lock is released will be visible after it is acquired, even if each operation happens in different threads. However, locks can slow the program significantly,

especially if they are used for long stretches of code or very often. An alternative approach is to use atomic accesses provided by the platform or programming language. These are faster than locks, but slower than unsynchronised access and can

While DIVINE is undoubtedly
not the only tool that can
analyze programs running
under relaxed memory,
we have shown that its
performance is comparable
to the best tools

only operate on a single memory area of limited size (e.g., 8 bytes on 64-bit platforms). Atomic accesses are often used to implement high-performance thread-safe data structures that can be accessed from many threads at once. If the programmer chooses to use atomic accesses, they will have to consider

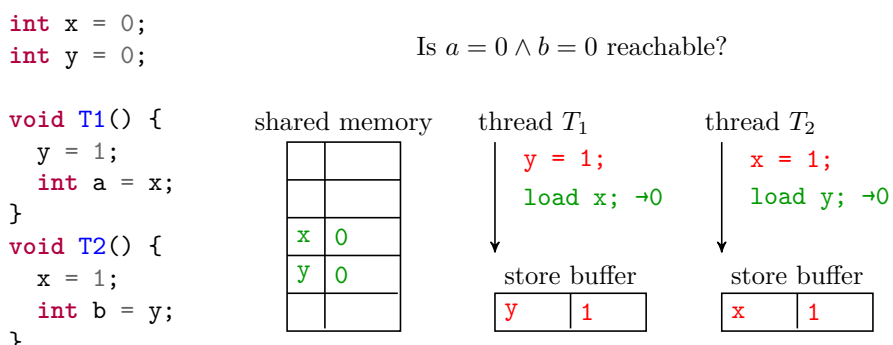
the possible ordering of events very carefully and always keep in mind that it does not behave as expected.

Furthermore, testing program behaviour under relaxed memory is especially hard. Not only do we have all the problems already mentioned for parallel programs, but an improperly synchronised program can also be very susceptible to minor modifications. For example, a tool that tracks memory accesses to detect unsynchronised parallel access can also easily mask relaxed behaviour and related errors.

In 2018, we published an extension of DIVINE¹ that allows it to analyze programs running under the x86-TSO memory model that describes relaxed behaviour of x86-64 processors and therefore should encompass behaviour of both current and future x86-64 processors. With this extension, DIVINE can find bugs caused by relaxed behaviour that would manifest on these processors.

While DIVINE is undoubtedly not the only tool that can analyze programs running under relaxed memory, we have shown that its performance is comparable to the best tools that handle x86-TSO and that each kind of tool seems to have different strengths and weaknesses (i.e. they can handle different kinds of programs well). We believe that DIVINE, with the wide range

Figure 1. A variable in the store buffer has not yet been saved in the memory.



¹Vladimír Štill and Jiří Barnat, "Model Checking of C++ Programs Under the x86-TSO Memory Model," DIVINE 4, divine.fi.muni.cz/2018/x86tso

of bugs it can detect and with good support for C++, can be a useful tool for analysis of thread-safe data structures.

DETECTING NONTERMINATING PARTS OF PROGRAMS [^LNTERM]

Another interesting problem in parallel software is termination. It often happens that one action waits until the other finishes. For example, getting an element from a thread-safe queue might wait for an element to become available, or entering a critical section must wait until another thread leaves it. Furthermore, we are often not interested in termination of the whole program. It might be a daemon or server (or its event loop in the test).


Research in this area is much less intensive than in the case of relaxed memory. Most approaches to termination focus on sequential programs, or on specialized modeling languages for parallel programs. Even the approaches that target parallel programs in realistic programming languages are often focused on the termination of the whole program.

In 2019, we published a paper about an extension of DIVINE that allows it to find nonterminating parts of programs.² To know which parts of the program must terminate, we use the notion of resource sections, essentially a piece of code with specified entry and exit points, such as a function or its part.

²Štill and Barnat, "Local Nontermination Detection for Parallel C++ Programs," DIVINE 4, divine.fi.muni.cz/2019/Interm

With these resource sections marked in the program, the extended DIVINE can check that the program cannot get stuck inside any of these resource sections. Some resource sections can be marked automatically by DIVINE. These include waiting for locks, critical sections of locks, waiting for condition variables, and waiting for thread joins. The user is also able to insert new resource sections in their code by simple annotations. For example, an author of a thread-safe queue with a blocking dequeue operation might want to mark it as a resource section.

WHERE IS DIVINE HEADED NEXT?

Research around the DIVINE analyzer also focuses on other topics, currently mostly on a symbolic and abstract representation of data. This allows DIVINE to handle programs in which some variables have arbitrary values or some inputs contain arbitrary data. DIVINE can then decide if the program is correct for all possible values of such data. These kinds of analyses come with a significant increase in computational complexity, and some of the contemporary research in our group focuses on ways to improve its efficiency through the use of lossy abstractions and their iterative refinement. Further research in our group includes enabling the decompilation of x86-64 binaries into LLVM, which would allow DIVINE to analyze native binaries. There is also research on the use of symbolic data in decompilation. However, the research into decompilation is in quite early stages. 

We believe that DIVINE, with the wide range of bugs it can detect and with good support for C++, can be a useful tool for analysis of thread-safe data structures.

Feature

Mental models: Qualitative research to design for Red Hat OpenShift users

To design effectively for our users, we need to learn more about them. If we don't, we may make a product that our users can't be efficient in, or worse, a product that our users have no need for in the first place. Our group within Red Hat's User Experience Design (UXD) team works specifically on the part of Red Hat OpenShift that operators (or system administrators) use to deliver the OpenShift platform on which developers create applications. We wanted to get a firmer understanding of who our users are. What do the users do every day? What motivates them? What challenges do they face? The operator's problem space is complex, especially to those without a background in system administration.

By Carl Pearson and Sarahjane Clark (Red Hat UXD Research), with Brian Dellascio (Red Hat UXD Visual)

One way that our UXD team can be sure we're starting with our users in the design process is by using a set of personas. A persona is a summarization of a segment of users or customers. Each persona segment typically gets a name, a picture, generic attributes (such as income and age), and highlights about their attitudes, needs, and beliefs.

The marketing field has used personas for decades in their work. For our UXD team, we needed to go even deeper into our users' backgrounds and work than a typical marketing approach. Because the UXD team designs all minute interactions within a visual interface, we decided to use a specialized persona creation process that

prioritizes day-to-day tasks of our users over high-level attributes and beliefs. The mental model approach by Indi Young (indiyong.com/examples) is a robust technique to visualize the complexity of a job like that of an OpenShift operator.

MENTAL MODEL RESEARCH PROCESS

The mental model approach uses qualitative interviewing to uncover nearly all tasks related to a user's main job (in this case, the job of delivering a cloud platform). A task in this case is simply something a user completes for their job that focuses on their goal and not the underlying technology used to do it (write documentation, create a cluster, send an outage message to a customer, and more).

The interview approach we used focused on exploration rather than confirmation. We aimed to uncover tasks that our users regularly complete that we didn't already know about. For this reason, our interviews were semi-structured: they had loose prompts but not a checklist of questions we asked. This allowed users to drive the interviews towards what they do in a naturalistic way. We interviewed nine OpenShift users, with a mix of titles and backgrounds, but all worked on OpenShift or Kubernetes platforms.

To analyze the data, we manually extracted every single task that was mentioned in an interview. We reduced hundreds of tasks to remove duplicates and ended up with over 200 tasks. Scanning a list of 200 tasks is nearly impossible to make sense of, so we grouped the tasks into "task towers" and "mental spaces." A task tower is a set of highly related tasks (platform health monitoring or scaling the platform). Task towers then fall into a broader grouping

of a mental space. This is higher level categorization, such as platform management. Ultimately, the final mental model map is a rich, organized set of all the tasks our users complete in their work. The mental spaces and task towers can be used to navigate the complex space that our users work in. **Figure 1** (below) shows our mental model map.

In this map, each card is a task. Each vertical stack is a task tower, and each gray horizontal bar is a mental space that groups tasks towers. This map was originally all gray until we incorporated our next step, personas.

The mental model map can be used in a number of ways by designers, product managers, or anyone who wants to explore the full context of users. One typical activity is to map existing product features vertically above each tower where a user goal is met. This can illuminate areas where new features may be needed. Another activity would

be a design thinking workshop, a group activity where designers and engineers create solutions to problems. In order to create good solutions, it helps to know where real problems are, and this map can help design thinking workshop participants understand where the problems might be.

The mental model map can be used in a number of ways by designers, product managers, or anyone who wants to explore the full context of users.

THE PERSONA PROCESS

The mental model alone doesn't segment the map into who may be more or less likely to spend time on each task. We also developed a set of personas to segment the tasks and clarify how certain users may vary across the map. This is critical for our UXD team because we know that OpenShift users have a variety of goals for typical tasks. We wanted to ensure we could properly design for unique use cases depending on the goals of certain groups of users, for example, developing platform features more often or monitoring platform operations more often.

Using themes drawn from the task towers and mental spaces, we created a set of behavioral variables (from Alan Cooper's *About Face*) that

Figure 1. The mental model map



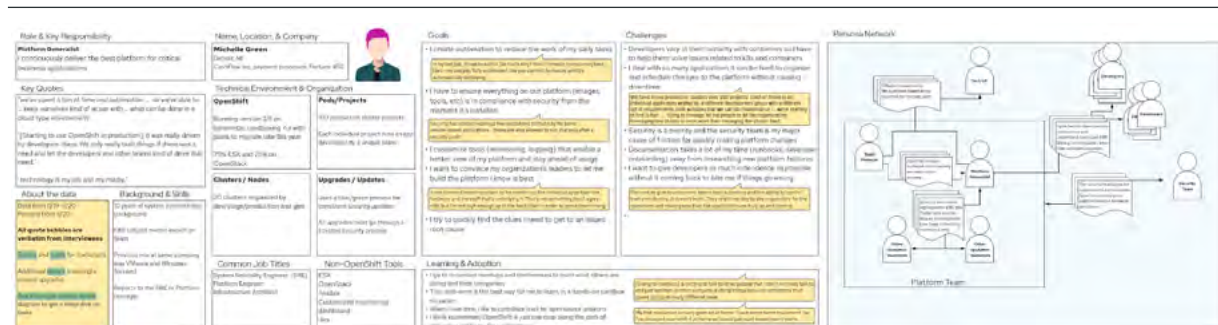
represented the spectrum of possible behaviors across our participants. We placed each participant on the variable spectrums. Then we qualitatively assessed how our participants clustered together. These formed the basis for how many personas we developed and what they represented.

Figure 2. Behavioral variable examples



After splitting up our participants into three personas, we dove back into verbatim quotes to form typical background characteristics, technical environments, goals, challenges, learning strategies, and team networks. We also coded the tasks from the mental model map to show which personas were more likely to complete each task.

Figure 3. Example layout from *The Generalist* persona




On the opposite page, we give a snapshot of each persona, but it's best to get into the full PDF to learn about all the details of a typical OpenShift operator.

PUTTING MENTAL MODELS TO WORK

We built this set of personas based on data from real users and a rigorous qualitative data analysis process. These personas haven't been quantitatively validated, but they're rooted in the real experience of those using OpenShift/Kubernetes. So far, the UXD team has used these results to jump start ideation work in the creation of new, innovative features for our OpenShift operator customers. The mental models and personas have also begun to serve as a platform for quickly bringing new designers up to speed in our highly technical space.

As with any good research project, our work opened up new questions. Our follow up is to explore a gap with an Operations persona, someone mostly concerned with monitoring and troubleshooting production clusters rather than building new cluster features. It's likely they will not have any fundamentally different atomic tasks to add to our mental model, but will result in a new persona based on how often they do certain tasks.

The mental model map and subsequent personas are a useful set of methods, especially for anyone designing in a technical user space. Our evergreen results speak to the broader nature of being a system administrator and will hold true for many years to come. 

THE BUILDER



This persona is most concerned with **initial building of the cluster** and how it will fit into a company environment (consultant is a common role).

Key goals: researching new distributions and features to create PoC clusters or initial configurations to hand off to operators.

Key challenges: working to support an operations team that may lack their OpenShift sophistication and having to keep up with documentation writing.

THE GENERALIST



This persona is who we may think of most commonly as an OpenShift operator. They are **capable of building initial clusters and being the expert in their operations** as well (SRE is a common role).

Key goals: automating everything possible to open up time for more platform development and to ensure their platform is developed and operated in compliance with the security organization's requirements.

Key challenges: ensuring developers are enabled to be successful on the platform and navigating scheduling with dozens of different teams using their platform during upgrades/patches.

THE MANAGER



This persona **leads a team of SREs** and sometimes dedicated operations engineers. They are tied closely to the business needs around their platform but also able to get hands-on in the UI/CLI when necessary.

Key goals: guiding their team to prioritize automation and to advocate for technology choices with VP level management.

Key challenges: keeping their high expertise SREs from having to spend too much time troubleshooting small issues and keeping costs balanced with optimal platform processes.

About the Authors



Carl Pearson joined Red Hat as a user researcher after earning his PhD in human factors psychology at North Carolina State University. He employs

a mixed-method research approach to uncover the core needs of OpenShift operators and how they use OpenShift.



Brian Dellascio is a principal user experience visual designer at Red Hat. He feels that beautiful design gets out of

the way and becomes a vehicle to disseminate large/complex data sets.



Sarahjane Clark is a senior user experience researcher at Red Hat, supporting Red Hat OpenShift. Her focus on finding deep insights through qualitative research studies, backed by solid quantitative

data, helps teams design and build products that make operators, system administrators, and developers successful in their work.

Interview

Open source cybersecurity and the next generation of computer scientists

**About the Author**

Chief security architect **Mike Bursell** joined Red Hat in August 2016 in the Office of the CTO, following roles working on security, virtualisation, and networking. After training in software engineering, he specialised in distributed systems and security, and has worked in architecture and technical strategy for the past few years. His responsibilities at Red Hat include security strategy, external and internal visibility, and thought leadership. He is one of the co-founders of the Enarx project (<https://enarx.dev>). He regularly speaks at industry events in Europe, North America, and APAC. He is currently writing a book for Wiley on Trust in Computing and the Cloud.

by Mike Bursell

Red Hat Research Quarterly invited Mike Bursell, Red Hat's Chief Security Architect, to chat with **Václav Matyáš**, Professor with the Centre for Research on Cryptography and Security at the Faculty of Informatics, Masaryk University, about his focus on cybersecurity education and how he has leveraged open source software projects as the laboratories and proving grounds for his students' work.

Mike Bursell: First of all, tell me about how you got involved with open source.

Václav Matyáš: That's a good question, and it goes well over ten years back in history. In the Faculty of Informatics [at Masaryk University], while working ad hoc with various industrial partners in Brno, we decided to put some form and coherent structure to our industry cooperation. So we surveyed the town, and since we had been keen on open source since our establishment, twenty-five years back, we found the most natural vibe with what we were doing and thinking in Red Hat.

We thought, okay, Red Hat is the place to discuss more systematic and even wider cooperation. We found the people that were keen to discuss it, like [Red Hat engineer] Radek Vokál and others, and it made sense, it fit, and it started working.

Mike Bursell: Excellent. Twenty-five years of open source, that's a long time.

Is that open source commitment a decision that your faculty made, or the university made?

Václav Matyáš: Another good question, and I don't know the answer. I can only speculate, and my speculation would be that it was in the blood and ideas that we started the faculty with, twenty-five years back. I was a PhD student then, and we always tried to have the publications open. We wanted to see the software working, wherever it was possible, and we wanted to get feedback with respect to that software.

Mike Bursell: You got involved with open source as a PhD student, or even before that?

Václav Matyáš: During my PhD studies. There were many more people within the Faculty of Informatics then that were already contributors to open source. I was just a user and observer.

Mike Bursell: It wasn't on my list of questions, but I'm very interested in open source and security, specifically. Do you have any strong feelings about the position of open source and security, or maybe the other way around, security and open source?

Václav Matyáš is a Professor at Masaryk University, Brno, CZ, acting as the Vice-Dean for Alumni Relations and Lifelong Learning at the Faculty of Informatics. His research interests are related to applied cryptography and security; he has published well over 170 peer-reviewed papers and articles and has co-authored several books. He worked in the past with Red Hat Czech, CyLab at Carnegie Mellon University, as a Fulbright-Masaryk Visiting Scholar at the Center for Research on Computation and Society of Harvard University, Microsoft Research Cambridge, University College Dublin, Ubilab at UBS AG, and as a Royal Society Postdoctoral Fellow with the Cambridge University Computer Lab. Václav also worked on the Common Criteria and in ISO/IEC JTC1 SC27.





Václav Matyáš: I do. I have a very funny story. There is a small association of industrial players that is hosted by the university, and it tries to play some role in cybersecurity education at secondary schools. They came out with a publication, a few weeks back, where they took a position that basically dates to the previous century: "open source is not reliable because

nobody is supporting it, and you cannot really trust it, so therefore it should not be used for security purposes." When I read that, I thought—okay, this must have been written by somebody who still wears the same glasses as in the 1990s.

So today we had a very nice discussion with a couple folks from the town, and also from Prague,

on openness, confidentiality, and open source—not only open source software, but also open hardware. It was very useful for us, because many of the discussions went back to factors that were true deep in the last century.

Today the situation is like this: You have an algorithm, mathematically proven, sound under various constraints, okay, nice, but then there is some logic with which you implement it, maybe even in some dedicated hardware. Then come the questions: Where did the hardware come from, what are the impossible vectors in that, what do you know about that hardware? Then you realize that you know much less about the hardware than you know about most of the software. Why would you scrutinize software so much in terms of algorithmic complexity, maybe even performance, and look at it from various angles, security being one of them, before you say okay, this software is good enough to run, but then you run it on hardware about which you know nothing? I am definitely a keen promoter of open insight into security solutions.

Mike Bursell: I was looking through some of the work you've been doing on your academic page, and I noticed a piece of work you did on the usability of OpenSSL. Tell me a bit about that, and why that's important.

Václav Matyáš: We came to that about four years back when we realized that OpenSSL has got well over twenty years of history, but for people who don't use it every day,

there's much too much to learn just to do basically primitive functions.

From that we thought, okay, let's examine this systematically. This was an area that started booming about ten years back, with end users and usability. We thought, everybody does that, so let's look at this from a different angle. We have users of the library that are developers and skilled IT folks, and we

...we realized that OpenSSL has got well over twenty years of history, but for people who don't use it every day, there's much too much to learn just to do basically primitive functions.

tell them to use OpenSSL for this and that. So they are using this famous security library. Then we'd measure what they like about it, what they don't like, what is easy to do. We did the first evaluations, and then we figured out that the scope was too broad, so we narrowed it down to the work with public e-certificates. Even just that will be a pretty good PhD thesis.

My student, Martin Ukrop—I believe he has an article in the last issue of *RHRQ* (<https://research.redhat.com/research-quarterly-2-2>, "Don't Blame The Developers" – Ed.)—was a very good and natural pick to be the PhD working on this, and with him we designed

the experiments. Of course, after the experiment, you will not only learn the results, but you will learn everything you did badly in designing the experiment, how it should be better next time. We repeat this in basically annual cycles.

Mike Bursell: It's not just how you create certificates, but how you use them and how they're trusted. Could you talk a bit about that? Because it underpins so much of what we do on the internet these days.

Václav Matyáš: Definitely. Certificates have been around for about two decades, but still a majority of people who are using them and relying on them misunderstand what they are good for, and how they work. Our experiments showed this clearly, even with the skilled people who are IT pros. They don't understand what the OpenSSL system is telling them about why a particular certificate is good for this, and not good for that. We realized it's actually a lot of stuff that goes beyond code, to documentation error messages.

Most of the research that we did in the past three years is now public, and it went through some nice publications. Now we're working on another, which will be Martin Ukrop's final paper, that shows our effort and improvements we did to redesign error messages and redesign documentation. We actually have very nice support from Red Hat staff. So the last paper will be showing positive outcomes: not only what the problems are with SSL, but what we could improve, what we did improve,

and where we still feel some debt to the project. We did not have the power to finalize the improvements, and we hope other people can follow up.

Mike Bursell: Do you often work with Red Hat when you find these sorts of things where improvements could be made? Obviously you're interested in not just pointing out problems but improving security. How do you work with Red Hat, and what other routes do you take in the faculty?

Václav Matyáš: Why and where with Red Hat? One reason is open source: we want to see the improvements happening, and as much any academician we want to see things working and be published.

With some companies that takes years. For me as a researcher, on one hand it's positive, on the other hand frustrating to give someone an idea that they don't implement, then four years later they decide that it's a very good thing and they use it in another project later on. Whereas in open source, you can see whether the thing is useful right now, and we can actually go forward and help.

We are not afraid of coding, so when we see that things can be adjusted in this way or that way, we can do that. With a company that has proprietary software, sometimes even under NDA for both sides, we cannot make this step forward. From this perspective, our work with Red Hat fills these academic expectations, and makes us happier to work with projects that are open source.

Mike Bursell: For me this is how open source should work. It's contributing not just code, but knowing you can contribute code, and audit code, and test, and make sure that the ideas and the theories are also all in there. I love how that fits at several layers.

I was looking at the other work you've done in crypto primitives. Could you talk about that?

Why and where with Red Hat? One reason is open source: we want to see the improvements happening, and as much any academician we want to see things working and be published.

Václav Matyáš: This comes from a series of projects that we had, both with the Czech National Security Agency of that time, and industrial partners. We found that we often work with computer systems, typically small hardware like smart cards, where we cannot check what they actually run, but we can observe through various side channels how they run things, whether they run things one way or another. Even if we do not know exactly how they run things we can watch the output, so we then became obsessed with watching output. We got a crazy idea: okay, we do not know how these designers of these hardware pieces—not just smart cards—do that, but let's have these

things running for weeks in clusters, and let's generate millions of keys out of that.

We later repeated the same effort with the crypto libraries, where it was much easier, but the inspiration came from closed hardware. Then we thought if we generate these millions of keys— it was actually tens of millions of keys ultimately—then we can run various checks, look for patterns. We figured out some things that were not expected at all: where people expected some particular data, particular components, or cryptographic keys to be random, we were able to show visually and numerically that clearly these are not random.

Mike Bursell: Explain to people who may not technically know as much about security why that's important, why the randomness of these things is important?

Václav Matyáš: It's important for a very simple reason. If you see somebody tossing a coin, and you see five times head, head, head, head, what do you expect the sixth attempt?

The mathematician will tell you the likelihood is still one-to-one. The naive person will tell you it's definitely going to be a tail. And the security person will tell you it's a biased coin, so it's definitely going to be a head again. You have the same situation, three different backgrounds, and three different answers to the question. The importance in security is best explained like so: Imagine that someone could choose a key that was created now or at any time

in the past, and your goal is to determine when the key was created. If you have no clue, and the chances of the key being created now or 50 million years ago are the same, then it's impossible to determine the time of the key generation. However, if it's not really random and the chances are not the same—for example you know that it has been generated today—then you definitely will not be guessing for the previous millions and tens of millions of years. You have a much more narrow subset of choices to make, and your chances to win this bet and find the right key are much higher.

Mike Bursell: The bet is basically that they can decrypt when you send your credit card to an online retailer, and they can maybe get your credit card because they have enough information to bet or to guess when it was encrypted?

Václav Matyáš: Exactly. It ties also to the non-crypto system security. If I have an unlimited number of guesses that only cost me some computing time, then I will try all the possible time expressions for today, and I will do what we call brute force it. If I know the key has been generated today, I can definitely guess the time if I'm not extremely limited in my number of choices.

But if the entire universe of time expressions was available ...

Mike Bursell: You'd be guessing for a long time.

Václav Matyáš: I can't do it.

Mike Bursell: Yes. Excellent. I wanted to talk about another thing, which I understand you're interested in, which is girls and women in tech. What do you do there?

Václav Matyáš: It's a loss of human talent, I would say. I've got three kids, and one of



...having a different angle and different view of the problem helps a lot, particularly in security.

them is a girl, or a woman now. I can see that when it comes to elementary school, and then sometimes even unfortunately high school, girls get the message, “Mathematics is not exactly for you, and computers, no, this is not for you at all. You will definitely find other subjects where you will excel better.”

Making these statements just based on sex is, a) of course, unfair, but b) it’s harming the entire human population, because from my experience women and girls have been successful in coding, successful in proposing computer systems, successful in history in running computer systems. Look at the World War II examples. I see no natural constraint that would make me claim there is some area of computer science where women are systematically worse than men.

I believe, given the different treatment of women in many ways, that at the age of productivity, after high school or university, the mindsets of women will be somewhat different than the mindsets of men, in some aspects. I’m generalizing now a lot, I understand, but having a different angle and different view of the problem helps a lot, particularly in security. If we deprive ourselves of this different look at the matter, then we are losing. We can have an attacker that may say, let’s have this group of people like women—or another group—looking into this, and they will see the mistake that we did, that we missed.

Mike Bursell: Indeed. I agree, this is very important. What does your faculty do? Do you do anything with Red Hat to encourage women and girls?

Václav Matyáš: I believe that I can say that we try to do a lot. Out of the universities in the wide region of the Czech Republic, when speaking of similar faculties or schools, our numbers are

impressive, but still not impressive enough for me. I mean we are now around 20 percent—sometimes we go to 22 percent, sometimes 19 percent—but around 20 percent of our intake to programs are women. This is better than other technical computer science and engineering faculties in the region, but it still can be improved significantly.

We hosted for many years the association that is fairly well known in Czech and Slovak Republics, Czechitas, the young ladies that actually have their primary mission to increase the number of women that are engaged in IT. Now they’ve gone to bigger offices than we can provide, but still we host a lot of their seminars. I was with them for some discussions just last week, when they had some trainings at our university, and we tried to cooperate with them as much as possible. We are meeting Red Hatters at these occasions quite a lot, actually, so thanks a lot to Red Hat for that.

Mike Bursell: If I were a person who’s not got a security background—I might be an undergraduate, I might be someone looking at moving in IT to something different—but I’m interested in security, where would you say I should start? Are there any books, are there any things I should read, are there any films I should watch?

Václav Matyáš: If it’s a university student already, then I would say grab a copy of Ross Andersen’s *Building Distributed Systems* book. For that you need the person to understand the basics of computer systems, and appreciate some factors of everyday life. This is not a book that I would recommend to a high school student, by any means.

For a high school student, then it actually depends. We do a lot of things with talented high school students, but these are students that were selected typically by their teachers of

computer systems, or they came to computer science from discussions with their parents or something. They already believe that computers are something that is maybe good for their future career. For these people, we already treat them nearly the same way as we do university students. We give them very specific tasks, we provide them with more guidance how to get to the solution, but still we do not provide much of a narrative how or why to do security.

Whereas, if we have discussions with general student groups in various high schools who are not these selected highly talented students, I would suggest probably another book where a lot of stories and interesting ideas come from, and it's tied to history: *The Code Book* from Simon Singh. These students will typically not be up to reading David Kahn. Giving them David Kahn is a very good option, but some may not read a book that thick. Singh is much shorter, and serves this purpose well.

Mike Bursell: I agree, it's a very good book. I sometimes tell people if they're looking for fiction and they want a big book then *Cryptonomicon* is a fascinating introduction to the world of how people think about it. There's lots of stuff about World War II and modern things as well, but it's a fun read, so that's another place I start.

Václav Matyáš: That's a very good read, too. You are right.

Mike Bursell: Can you give me some

examples of the things that you're doing in the faculty at the moment that you might point someone at?

Václav Matyáš: One of the things is a set of tools. It ties to the crypto primitives and helps one to check whether the crypto primitives, whether in hardware or software, have been implemented correctly. We have been redesigning many of them for quite a few years, providing them of course as open source and supporting them as well. This is definitely something that we believe in. The ultimate expectation is that you will have a semi-automated system to which you can give an implementation, whether in software and maybe in the future hardware as well, and it will not only tell you whether the functionality, let's say output of cryptography, is wrong or right, but if it's wrong it will give you a few hints as to what may have been the causes.

This is challenging. It will probably take a decade or two, but I believe that in the future these will be the first steps of cryptoanalysis. The way we do cryptoanalysis now, we do various tests, then we check the outputs with NIST and so on. In the future we are running pieces of software, and there is already enough knowledge to see that not only we run pieces of software, but that software—if we have reached enough data—tells us that there has been something fishy, and maybe why it has been fishy.

Mike Bursell: Yep, yep.

Václav Matyáš: So that's one. Another is the one we already discussed, and that's usable security. There we can document for the student, "This is the problem with public e-certificates, about which we told you about in class. These are the experiments that we did—you can easily read about them—and this is what we do now improving error messages." It's not just looking out of the window and just by the weather deciding how to redesign it; there is some systematic effort that we learn from psychologists how to work with people, how to experiment, and how to get to a better version. Similarly, I'll go through about three or four other areas.

Mike Bursell: Another question I was going to ask is, let's assume that I'm a talented MA student, or undergraduate student, and I come to you and ask which areas are going to be important in security in the next ten to twenty years? Which subjects would you say will be really interesting?

Václav Matyáš: That's a good question. We usually are very pragmatic. I mean you have two sorts of people. One sort comes with their idea and their problem already, and you have to nurture them, help them.

For example: We have a new PhD student that will be supported by Red Hat, starting in September. When I first met him about four or five years ago, he was an undergrad student, first year, and he came with a very interesting



DEVCONF.cz
open source community conference

February 2021
📍 **All around the world.**


virtual event

www.devconf.cz

Sponsored by  **Red Hat**

challenge. He said, "I am from this city in Slovakia, from Žilina, and we managed to break into their transport ticketing system. We reported the problems to them, and what we heard back was threatening by lawyers, and that we should not disclose this error to anybody."

I told him okay, I will help, but you have to pay for it, and you have to pay by working on the problem, describing it in detail, considering other angles and variants of the attacks, and documenting it. I took this student for a graduate course, where he wrote this report, and then we provided the report to the city transport company, giving them two months' leeway to have their problems fixed before the report goes public. As a report of a student it enjoys academic freedom under our laws, so it could be made public. Two months' time was more than sufficient to fix these mistakes, and by supporting this student in that particular scope of work, he now enjoys a lot of things that we do in the lab, and now he's working on other things that we already see as very good prospects for the future.

The other type of student just comes with open hands, and asks the way that you asked me. Then we say okay, we have these projects that we are running in the lab, and by running them it means that we are spending our effort on them, and that means that we believe that they will be useful. You see us and our people with whom we work at the university spending our time on this, so we suggest that you consider one of these things that we are doing. 

```

1995 const fetchingBlueprints = (state = false, action) => {
1996   switch (action.type) {
1997     case FETCHING_BLUEPRINTS:
1998       return true;
1999     // We went from selling boxes of Linux®
2000     case FETCHING_BLUEPRINT_NAMES_SUCCEEDED:
2001       // if 1 or more blueprints, fetching is true because we're still waiting
2002       on the contents)
2003       return action.payload.blueprints.length > 0;
2004     case FETCHING_BLUEPRINTS_SUCCEEDED:
2005       // to operating servers around the globe
2006     case BLUEPRINTS_FAILURE:
2007       return false;
2008     default:
2009       return state;
2010   }
2011 };
2012 const errorState = (state = null, action) => {
2013   switch (action.type) {
2014     // to giving new life to virtual machines
2015     case FETCHING_BLUEPRINTS:
2016     case FETCHING_BLUEPRINTS_SUCCEEDED:
2017       return null;
2018     case BLUEPRINTS_FAILURE:
2019       return action.payload.error;
2020     // to pioneering a new era in containers
2021     default:
2022       return state;
2023   }
2024 };
2025 // to powering over 90% of the Fortune 500*
2026 const blueprintList = (state = [], action) => {
2027   switch (action.type) {
2028     case CREATING_BLUEPRINT_SUCCEEDED:
2029     return [
2030     ...state.filter(blueprint => blueprint.present.id !==
2031     action.payload.blueprint.id),
2032     {
2033     past: [],
2034     // to bridging every kind of cloud
2035     present: Object.assign({}, action.payload.blueprint, {
2036     localPendingChanges: [],
2037     workspacePendingChanges: []
2038     }),
2039     future: []
2040     }
2041     ];
2042     // And we were able to do all this because

```

Our code is open_



redhat.com/ourcode

Copyright © 2020 Red Hat, Inc. Red Hat and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.
 *Red Hat client data and Fortune 500 list for 2019.
 Code licensed under the MIT license. <https://opensource.org/licenses/mit>.

Feature

A thread model for the real-time Linux kernel

The recent advances in AI and telecommunications are enabling a new set of complex cyber-physical systems, including those for safety-critical applications. Safety-critical systems are systems whose failure can result in significant damage, including loss of life. This class of systems ranges from medical devices up to advanced driver-assistance systems (ADAS) for vehicles. Some of these advances rely on a sophisticated software stack, requiring full-featured operating systems such as Linux. Among the features enabling Linux in such environments are the real-time Linux kernel features.

**About the Author****Daniel Bristot de**

Oliveira is a principal software engineer at Red Hat, working in the development of real-time features of the Linux kernel.

Daniel has a joint PhD degree in Automation and Systems Engineering at UFSC (BRA) and in Embedded Systems at the Scuola Superiore Sant'Anna (ITA).

by *Daniel Bristot de Oliveira*

This is the first in a series of three articles about the formal analysis and verification of the real-time Linux kernel.

Real-time systems are computing systems whose correct behavior depends not only on logical behavior but also on timing behavior. For example, the detection of an obstacle in an autonomous vehicle should result in a set of actions that need to be taken in time to avoid a collision.

Linux was not designed as a real-time operating system (RTOS) from scratch. Instead, it was adapted to become one. Nowadays, Linux has a set of advanced RTOS features. For example, it can schedule tasks using an advanced deadline-based scheduler (SCHED_DEADLINE) and react to external events within fewer microseconds with the PREEMPT_RT

patchset. Such features enabled a vast set of applications, from high-frequency trading systems to low-latency network communication.

However, the path taken by Linux developers in the analysis of the system differs from the method used by real-time researchers. The challenge for real-time researchers is to demonstrate that the coordinated behavior of the system produces results for all tasks before their respective deadlines, in the worst case. A common approach for such a demonstration starts with the analysis of the specification of the system and the formal definitions of the system properties. These properties are then translated into a set of variables that are used on mathematical analysis of the response time of the tasks of a system.

The evaluation of the real-time features of Linux took a more experimental approach: developing

tools to simulate real-time workloads, measuring the response time for their requests, and not explicitly evaluating the internals of the OS kernel. The problem with such an approach is that it is not enough to provide any strong evidence that the worst-case scenarios were found. Moreover, the absence of a formal analysis of the timing behavior of Linux, in the terms used in the real-time scheduling theory, is a challenge for the application of Linux on safety-critical systems where such sophisticated analysis is required.

The reason Linux developers use this approach has its roots in the Linux kernel complexity. The amount of effort required to understand all the constraints imposed by the synchronization mechanism on real-time tasks on Linux is not negligible. The understanding of the synchronization primitives and how they affect the timing behavior of a thread is fundamental for the definition of Linux in terms of real-time scheduling. The challenge is then to describe such operations using a level of abstraction that removes the complexity of the in-kernel code, and to do this in a format that facilitates the understanding of Linux dynamics for real-time researchers, without being too far from the way developers observe and improve Linux.

FORMAL MODELING

A model is an abstraction of a system. The process of modeling a system involves the definition of a set of measurable variables associated with

the given system. The subset of variables acting on the system from outside are considered input variables, while the subset of variables that are possible to measure while varying the input are defined as the set of output variables, as in **Figure 1**. The modeling phase of a system also comprises the definition of the mathematical relationship between the input and the output.

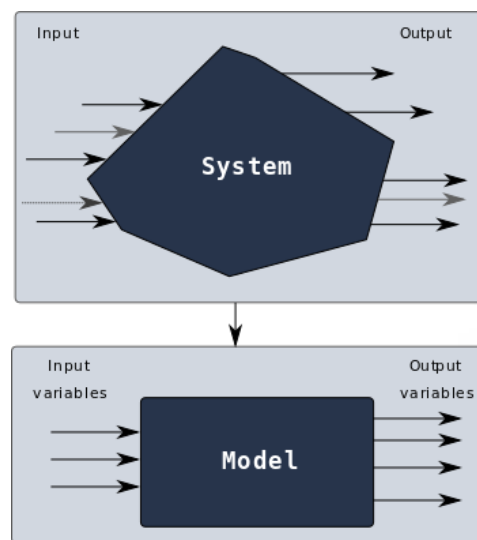


Figure 1. System and model

The use of mathematical notation removes the ambiguous nature of natural language and enables the application of a more sophisticated analysis of the runtime behavior of Linux. The problem is, which mathematical method could be used to model the Linux behavior?

The developers of Linux observe and debug the timing properties of Linux using the tracing features present in the kernel. For example, they interpret a chain of events, trying to identify

the states that cause delays in the activation of the highest priority thread, and then try to change kernel algorithms to avoid such delays. The notion of events, traces, and states used by developers is common to Discrete Event Systems (DES), which present automata as a modeling formalism.

The automata formalism is well established as a language for the modeling and verification of systems. Automata are characterized by the directed graph or state transition diagram representation, as in **Figure 2**, where the arcs represent the events, and circles mean the states. This simple format hides the complexity of the mathematical definition, which allows the use of sophisticated operations and analysis while allowing an intuitive way for reasoning about the property being specified, close to the way that the kernel developers already use while analyzing the traces of the system.

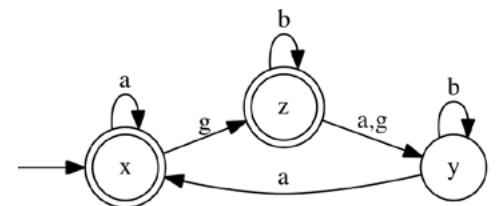


Figure 2. Automaton example

MODELING LINUX'S THREAD BEHAVIOR

To enable the use of a more sophisticated analysis of the timing behavior of Linux tasks, we proposed an automata-based model for Linux threads (which are the Linux task's

entities). The model aims to remove the code complexity of Linux by presenting a simplified view of the system, but still using the same set of abstractions used by kernel developers, enabling efficient communication between kernel developers and real-time researchers.

The approach used in the PREEMPT_RT Thread model development is presented in **Figure 3**:

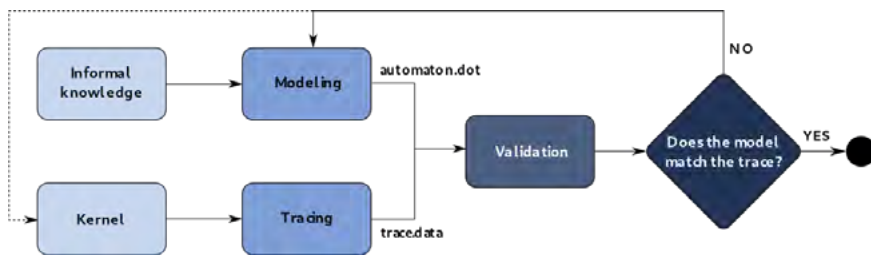


Figure 3. The modeling approach

The informal knowledge about the timing behavior of the Linux tasks was modeled as a set of formal specifications using the automata theory. The model was built upon a set of events used by kernel developers to analyze and describe the evolution of Linux threads. These events can be observed and analyzed using the Linux kernel tracing features.

The final version of the model was composed of 34 events, 9,017 states, and 20,103 transitions, demonstrating how complex the timing behavior of Linux threads is. However, the model was not built as a single monolithic automaton. Instead, the model was created using a modular approach, in which the final model is composed of the synchronization of a set of small models. These small models are divided into two classes. The *generators* represent the independent actions of the system. For example, *IRQs* can be *disabled* and *enabled* (**Figure 4**), and the

scheduler can be called and returned (**Figure 5**). The *specifications* describe the coordinated behavior of the generators. For example, the model presented in **Figure 6** specifies that the scheduler cannot be called while interrupts are disabled.

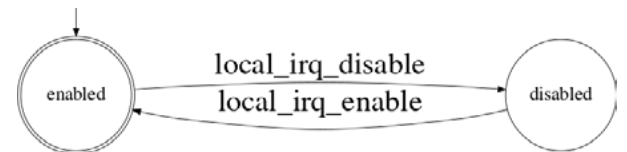


Figure 4. generator: irq disabled and enabled

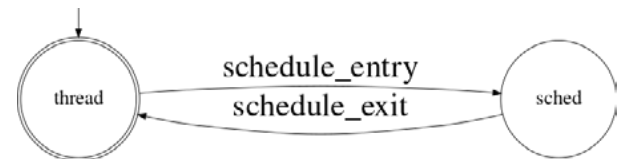


Figure 5. Generator: scheduler generator

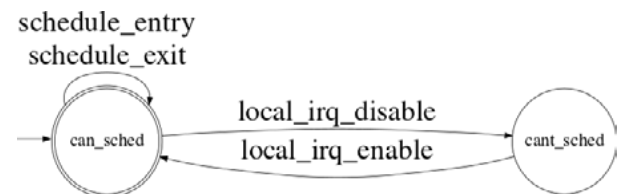


Figure 6. Specification: cannot schedule while IRQs are disabled

The final model is composed of 12 generators and 33 specifications. The vast majority of the generators and specifications are modeled with only two or three states, while the largest has only eight states. This is an essential factor: in the end, the complexity of Linux is indeed composed of a set of small specifications.

MODEL VALIDATION

The validation of the model was done using both the analysis of the properties of the automata, as well as a comparison of the model against the trace of the execution of the system.

The automata format allowed the analysis of the non-functional properties of the model. For example, the thread model is deterministic, meaning that one event can lead the system to a single conclusion. The model is free of deadlocks and livelocks, and it is possible to reach a safe state (in this model, the initial state) from all states of the system.

One of the main benefits of using the common event abstraction is that it facilitated the automatic validation of the model. During the development of the model, the *perf* tracing tool was extended to enable the execution of the automaton by using the events generated by a real implementation of the system. Initially, the tool pointed to many cases that were not initially covered by the model. However, at a given point, the tool started to unveil points in the Linux kernel code that were not following the specifications. Such cases were analyzed and reported to the kernel community, which confirmed three bugs in the kernel, evidencing the model's adequacy.

USAGE OF THE MODEL

The model has found two main applications: the runtime verification of formal specifications and the timing analysis of the Linux kernel.

The discovery of kernel bugs using the automata-based models motivated the usage of formal specifications for the runtime verification of the kernel. Indeed, the specifications presented in this research were later used as the basis for the development of an efficient method for the formal verification of the Linux kernel.

The model was also used as the base for the formal definition of the real-time Linux scheduling latency components, using the same mathematical approach used in the real-time scheduling theory, allowing a new set of timing analyses for the real-time Linux kernel.


FINAL REMARKS

Linux is a sophisticated real-time operating system and is enabling the development of a new set of cyber-physical systems, many of them with safety-critical and real-time requirements. Such a class of systems requires the application

of sophisticated analysis that evidences the correct behavior of the system, both in the logical and timing perspectives. Automata-based model usage allowed the formal specification of an intricate part of the Linux kernel, enabling the unambiguous understanding of the system behavior from the real-time systems theory perspective and the runtime

verification of the adherence of the kernel code to the expected behavior. The details of the analysis enabled by the automata model will be the subject of future articles in *RHRQ*.

This research was done in a collaboration of Red Hat with Prof. Rômulo Silva de Oliveira (Universidade Federal de Santa Catarina) and Prof. Tommaso Cucinotta (Scuola Superiore Sant'Anna).

More details of this research can be found in this paper: Daniel B. de Oliveira, Rômulo S. de Oliveira, and Tommaso Cucinotta, "A Thread Synchronization Model for the PREEMPT_RT Linux Kernel," *Journal of Systems Architecture* 107 (2020). DOI: 10.1016/j.sysarc.2020.101729. 

The model has found two main applications: the runtime verification of formal specifications and the timing analysis of the Linux kernel.



Project Updates

Research project updates

Faculty, PhD students, and US Red Hat associates in Israel are collaborating actively on the following research projects. This quarter we highlight collaborative projects at Technion University, Tel Aviv University, and The Interdisciplinary Center Herzliya. We will highlight research collaborations from other parts of the world in future editions of the Research Quarterly. Contact academic@redhat.com for more information on any project described here.

PROJECT: OpenCEP: An Advanced Open Source Complex Event Processing Engine

ACADEMIC INVESTIGATORS:
Prof. Assaf Schuster (Technion)

RED HAT INVESTIGATORS:
Ilya Kolchinsky

In *Red Hat Research Quarterly 2:1*, investigators described their plan to build a scalable, real-time, cloud-based CEP engine capable of efficiently detecting arbitrarily complex patterns in high-volume data streams. The engine was designed to be implemented on top of Red Hat® OpenShift® Container Platform and to be applicable to any domain where event-based streaming data is present. Researchers aim to create an open source project/community based on the engine. They also hope to advance the state of the art in the area of complex event processing and combining academic research with the implementation and deployment of novel CEP mechanisms and techniques in the above framework.

Now the first version of the OpenCEP, an advanced open source complex event processing framework with cutting-edge pattern detection capabilities, is officially available for use. The next development iteration is underway and is expected to end in January.

PROJECT: Kubernetes Optimized Service Discovery Across Clusters

ACADEMIC INVESTIGATORS:
Prof. Anat Bremler-Barr and Daniel Bachar, The Interdisciplinary Center Herzliya

RED HAT INVESTIGATORS:
Mike Kolesnik

This research project aims to provide better and more balanced service discovery capabilities for Kubernetes multi-cluster deployments. Currently, the service discovery in this space is very basic. The project aims to investigate and assess different approaches for improving it by making it more balanced, reducing bottlenecks, and improving latency.

PROJECT: **Electroencephalography (EEG) Feature Extraction**

ACADEMIC INVESTIGATORS:

Prof. Nathan Intrator (Tel Aviv University)

RED HAT INVESTIGATORS:

Boris Odnopozov

This research is meant to improve the treatment of electrical status epilepticus (ESES). ESES is an age-related epileptic encephalopathy (EE) with very low incidence, typically in individuals between a few months of age and 12 years, peaking around ages 4-5.

In ESES, there is a disorder in the epileptiform activity in the brain that takes place during sleep, causing "unseen" (subclinical) seizures, although clinical seizures can also occur. While this condition tends to resolve with time, there can be sequelae in neuropsychological development (e.g., language capacity, intellectual level, memory, behavior) as well as motor impairment. Treatment is aimed at controlling both the seizures and their cause, the epileptiform activity.

One of the challenges physicians face when treating this disease is the difficulty of seeing if medication is working properly. Currently this requires monitoring a child during sleep using a full set of electrodes, which is both expensive and burdensome. As a result, physicians do not have enough feedback on the efficacy of the medication, which makes treatment less effective. Using Open Data Hub (opendatahub.io) for

our calculations, we will attempt to see if we can detect ESES using only frontal electrodes. This allows a much easier, and more efficient home-based monitor that permits physicians to better administer medication and make treatment more effective.

For the past few months, we have researched the data for ESES detection with a reduced number of electrodes. We have added sleep-detection related features such as slow waves and spindles with the hope of differentiating sleep afflicted by ESES from normal sleep. We also added the use of UMAP to our pipeline for dimension reduction. These steps improved the precision of the classification.

We are faced with several challenges that we will take on in the coming months. First, the data we have is labeled at the recording resolution, *i.e.*, each recording is either labeled as ESES or not. ESES by definition may only show on the EEG recording 80 percent of the time. Therefore we need a more granular labeling. To that end, we need to find a way to display the false positives and false negatives so that physicians can manually assess and classify the recordings. This is not as simple as one might think, since physicians are used to working with specific medical software that presents data in a very specific way. We need to improve the pipeline and engineer more features to improve detection, and we want to use the full power of working with hyper-parameterization that Open Data Hub offers.

PROJECT: Ceph: Wire-Level Compression-Efficient Object Storage Daemon Communication for the Cloud


ACADEMIC INVESTIGATORS:

Prof. Anat Bremler-Barr and Maya Gilad,
The Interdisciplinary Center Herzliya

RED HAT INVESTIGATORS:

Josh Salomon

This project's purpose is to reduce storage network traffic (object, block, etc.) for the following cases: between the failure domains in cost-sensitive environments such as public clouds, and between nodes in cases where the network bandwidth is the bottleneck of the node performance. We have divided the project into three milestones: applying compression for data transfer between different datacenters, enabling/disabling compression given hints from the client, and minimizing compression efforts when data is non-compressible.

Maya has completed the coding and basic testing of the on-wire compression for Ceph. It was presented in the IDC demo day for the course, and it was one of only two projects that created real production code (other projects were more research projects, mostly testing new AI/ML models). The PR is now in the review process by the upstream community, and it seems it will be approved soon, after some minor fixes. Initial results seem very promising, and a detailed report with the results will be published once the PR is approved. 



AI ON INTEL®



**NOW BUILD THE AI YOU WANT
ON THE CPU YOU KNOW.**

Learn more at ai.intel.com