

# RH RQ

Bringing great research ideas  
into open source communities

BigDataStack  
delivers

User authentication



Faster hardware  
through software

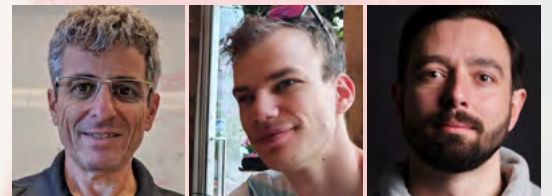
## Anna Brunström

the right idea at the right time



Red Hat  
Research Quarterly

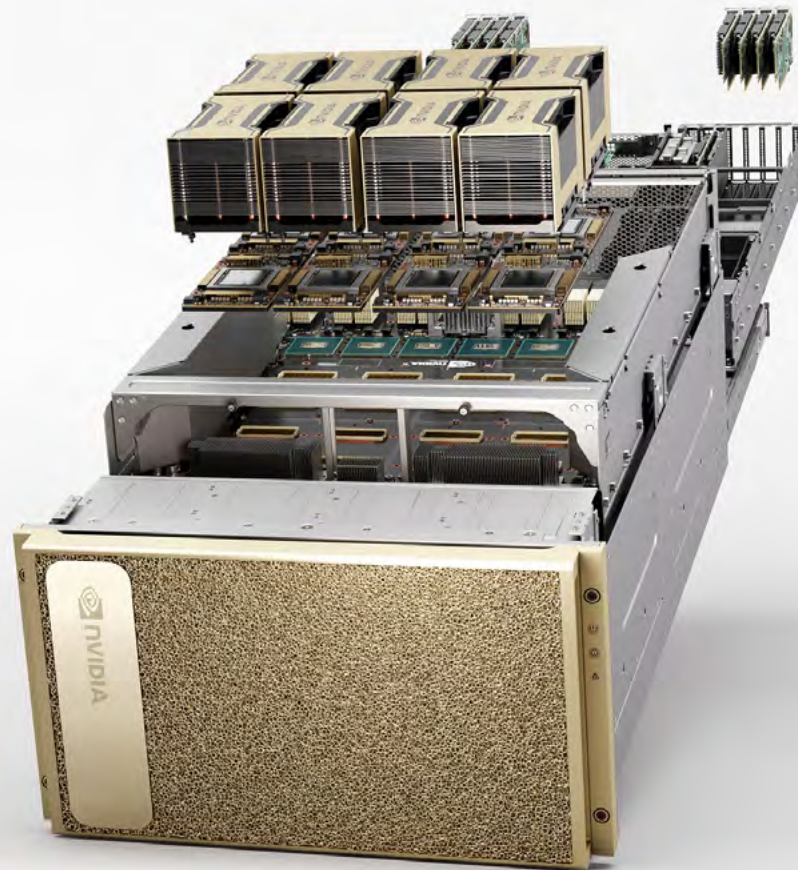
Volume 3:2 | August 2021 | ISSN 2691-5278



**Making the most of  
research mentorships:**

the building blocks of productive  
industry-university relationships





# THE UNIVERSAL AI SYSTEM FOR HIGHER EDUCATION AND RESEARCH

## NVIDIA DGX A100

Higher education and research institutions are the pioneers of innovation, entrusted to train future academics, faculty, and researchers on emerging technologies like AI, data analytics, scientific simulation, and visualization. These technologies require powerful compute infrastructure, enabling the fastest time to scientific exploration and insights. NVIDIA® DGX™ A100 unifies all workloads with top performance, simplifies infrastructure deployment, delivers cost savings, and equips the next generation with a powerful, state-of-the-art GPU infrastructure.

Learn More About **DGX** @ [nvidia.ws/dgx-pod](https://nvidia.ws/dgx-pod)

Learn More About **DGX on OpenShift** @ [nvidia.ws/dgx-openshift](https://nvidia.ws/dgx-openshift)

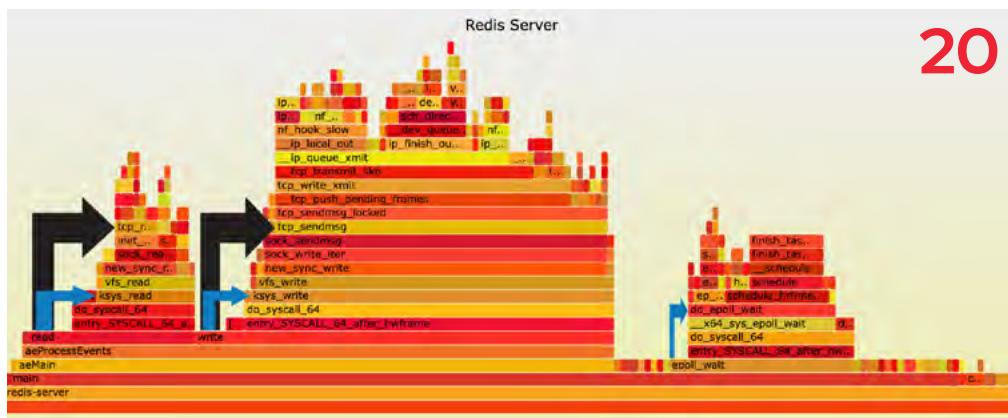
## Table of Contents



12



06



20



**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.

**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



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

### Departments

- 04** From the director
- 06** News: Europe RIG launch
- 06** News: Red Hat joins IBM Fellowship program
- 07** News: Engineer gender diversity on the rise
- 36** Research collaboration shaping the future
- 38** Research project updates

### Features

- 08** User authentication in public open source repositories
- 12** The right idea at the right time: an interview with Anna Brunström
- 20** Faster hardware through software
- 25** Making the most of technical research mentorship
- 30** BigDataStack delivers

**From the Director**

# Programmable networks, hardware— what's next, programmable enterprises?

*by Hugh Brock***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.


I don't read a lot of business books. I don't usually find them well written or entertaining, and the lessons they hold are—well, if you've gotten far enough in your career, you ought to be able to figure this stuff out for yourself, right?

An exception to this is Peter Drucker. In his 1954 classic *The Practice of Management*, he exhorts managers to push decision-making as close to individual workers—and as near to the last minute—as possible, an idea as fresh today as when he wrote it. A self-aware manager knows they must leave as many decisions to their associates as possible, even risking failure, because the payoff in their team's effectiveness is so great. But it turns out this is hard, so most of us fall back on giving orders we hope are correct.

What does this have to do with programmable networking? In this issue's interview with Karlstad University professor Anna Brunström, Toke Høiland-Jørgensen talks with her about the astonishing performance gains she has found using programmable networking on general purpose hardware. Allowing a programmer with application knowledge to code what should happen to the packets that application is concerned with is far more efficient than having a switch designer try to anticipate all possible cases in advance. As with management, pushing networking decisions closer to the programmer is the most efficient way.

We find a similar theme in programmable hardware in Gordon Haff's review of our Boston University

research projects. Processor makers have spent years and billions of dollars building optimizations to try to do the software developer's job for them. Using extra cycles and circuit paths, a modern processor makes great speed gains by guessing what code is going to do in advance and invoking optimized paths to do those things. The results are impressive but expensive in power consumption and processor transistor count. As we approach a time when datacenters will consume more than twenty percent of all generated electricity, we need to find ways to be more efficient. Haff walks through a set of projects designed to do computing more efficiently by making optimization decisions dynamically as they are needed, rather than years in advance in the processor design phase.

Good decisions depend on good information, however, and our piece by PhD student Agáta Kružiková and Red Hat security authority Milan Brož shows that even cautious developers aren't as careful as they should be with security. Kružiková surveyed developers at a large engineering office in the Czech Republic to find out how careful they were with user authentication for their GitHub accounts. Although the results were generally encouraging, it was disappointing to see the number of developers who still do not use the strongest possible authentication. More work on communicating the risks of compromised accounts, and making authentication usable, is clearly necessary. 



# Red Hat Research Days 2021

A monthly series of conversations between researchers and Red Hat experts



**Ilya Kolchinsky**  
Red Hat, Technion



**Oren Oichman**  
Red Hat



**Petr Švenda**  
Masaryk University

**"The Future of Big Data: Massive Real-Time Data Streams"**

**"Mining Issued Common Criteria and FIPS 140-2 Certificates"**



**Martin Herbordt**  
Boston University



**Robert Munafo**  
Boston University



**Ahmed Sanaullah**  
Red Hat

**"A Plan for Practical Programming of FPGAs in the Data Center"**



**Daniel Gruss**  
TU Graz



**Moritz Lipp**  
TU Graz



**Jon Masters**  
Red Hat

**"Platypus: Power Side Channels in Software"**

Watch on demand  
[research.redhat.com/research-talks](https://research.redhat.com/research-talks)

## News

**About the Authors**

**Matej Hrušovský** has been with Red Hat for more than eight years, six of which have been spent managing the university program in EMEA. Aside from attracting new talent mainly from universities and schools, the core of Matej's job is to find and put the right people from Red Hat and academia in the same room together.

## Europe RIG launched June 3, 2021


*by Matej Hrušovský*

**R**ed Hat Research's Brno Research Interest Group (RIG) and the established Red Hat Research Centre at Newcastle University have agreed to join forces and form the Europe RIG. Launched June 3, 2021, the new RIG will combine these long-lasting programs and many smaller ones into a much bigger roundtable to enable greater collaboration and more and better projects. This should result in better allocation of resources and better cross-regional coordination when applying for long-term research projects and grant programs in the European Union.

Red Hat partnerships with universities in Europe have been around for more than a decade. In Brno, Red Hat's largest engineering office has benefited from new talents developed at several regional universities for almost fifteen years. The Red Hat Research Centre at Newcastle University, begun in 2010, supports long-term research into future technologies. The new RIG will also support projects on advanced



networking at Karlstad University in Sweden, side-channel vulnerabilities at TU Graz in Austria, and other projects throughout Europe.

In addition to the new Europe RIG, there are RIGs in Israel and in the greater New England area of the US. Researchers and engineers who want to be involved in RIG activities are welcome; visit [research.redhat.com](https://research.redhat.com) for more information. 

## Red Hat joins IBM PhD Fellowship program

*by Marek Grác*



**Marek Grác** is a senior software engineer at Red Hat.

**T**his year, Red Hat has become a part of the [IBM PhD Fellowship program](#), an award program that recognizes PhD students who demonstrate both academic excellence and exceptional innovation in their research proposals. The 2021 program received hundreds of applications from 183 universities in 32 countries.

As part of our participation, Red Hat experts helped select the best candidates from a pool of applicants as part of the cloud and machine learning panel. Three candidates in this area were chosen, focusing on three topics: improving system performance, clean-slate operating systems in Rust, and federated learning.

The benefits of the IBM PhD Fellowship Awards include funding for PhD students in the final years of their programs, as well as access to mentors from both IBM and Red Hat, in addition to their academic lead. For example, Red Hatter Uli Drepper will be working with University of California, Irvine student Vikram Narayanan, whose recent work in RedLeaf, a clean-slate operating system in Rust, explores the possibility of abandoning hardware-based isolation in favor of language-based mechanisms—thereby changing the course of operating system design.

The IBM PhD Fellowship program is an intensely competitive worldwide program focused on PhD students looking to make their mark in promising and disruptive technologies. Students must be nominated by a doctoral faculty member. Nominations for the annual IBM PhD Fellowship program begin the third week of September and are accepted for five weeks. Find out more at [research.ibm.com/university/awards/fellowships.html](https://research.ibm.com/university/awards/fellowships.html). 

## Gender diversity on the rise among engineers


by Matej Hrušovský

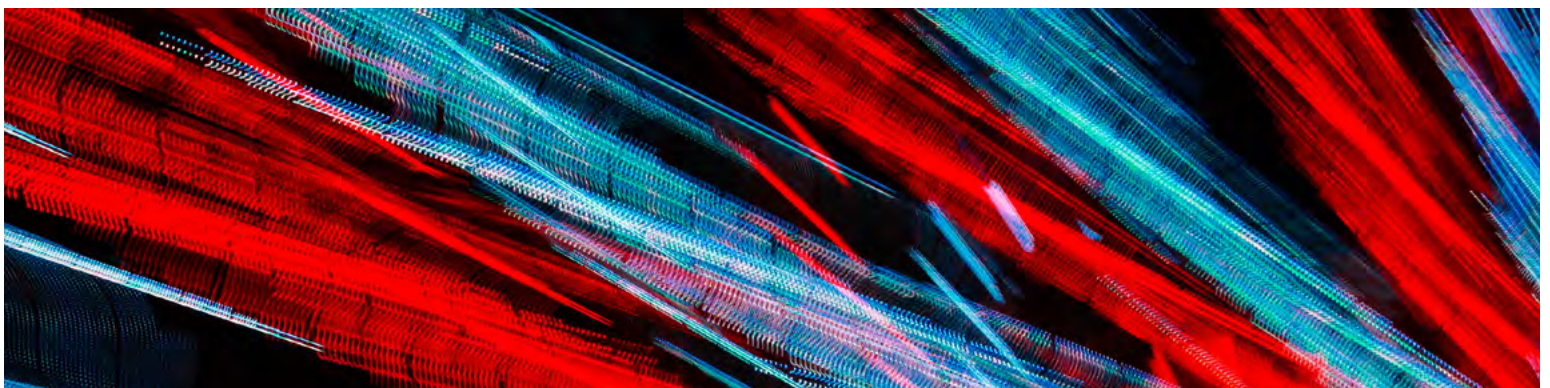
**M**uch of the focus of IT industry diversity efforts has been at the university level and beyond. At Red Hat Czech, however, a systematic focus on mentoring in high school and even grade school is paying real dividends in gender diversity in our hiring.

Like others, Red Hat initially focused its academic diversity efforts on developing a broader talent pool in universities, supporting university-organized diversity and inclusion events aimed at uprooting the attitude that women and STEM don't mix. For example, Red Hat supports the Czechitas Thesis Award ([czechitas.cz](https://czechitas.cz)), providing mentorship for young women passionate about working in IT.

However, influencing diversity at the university level may be too little too late.

That's a key reason we started our high school programs. The stigma attached to women in STEM is even more prevalent on the high school level, and that's where the most work is needed. At events like the Brno University of Technology's (F)IT Summer School for Girls ([holky.fit.vutbr.cz](https://holky.fit.vutbr.cz)), Red Hat has provided mentoring for high school students by demonstrating what working in IT entails and showing why women have an important place in the equation.

Increasing diversity and inclusion is not a magic trick where we can conjure results with a snap of a finger. It takes a while to see demonstrable results. And it's likely that change cannot be linked to one cause. Rather, it is driven by a combination of sustained efforts and early intervention, rather than waiting until it may be harder to drive change. 





## Feature

# User authentication for open source developers: what do they use?

Ongoing research into user authentication in public open source repositories demonstrates the importance of usability—even for IT professionals.

*by Agáta Kružíková and Milan Brož*

**O**pen source software companies build products on the work of open source developers and free content creators. Products based on open source development may depend on hundreds of projects whose upstream developers have no relation to that company or its products. While we have many tools that provide code analysis, license tracking, and continuous integration—in keeping with today’s focus on the whole development supply chain—do we know who the independent developers upstream are and how they behave?

For this project, we focused on a simple piece in upstream development: authentication of independent developers on GitHub. No company policy for authentication binds these developers; no single sign-on is in place. How and if secure authentication

will be used is entirely based on the project developers themselves.

Misusing a developer account can end in severe damage, and focusing only on detecting such incidents would be costly (and a response would come too late). There are many developers whose activity changes over a time period. For example, students may be active in some projects, focusing on new, exciting ideas, or developers may contribute only once. An upstream maintainer of a security-related project sees random attempts to get access to projects from unknown people or attempts to log in under existing developer accounts quite often. Strong authentication is crucial to prevent possible project tampering, which is inevitably followed by a loss of trust.

---

## About the Authors

**Agáta Kružíková** is a PhD candidate at the Centre for Research on Cryptography and Security at Masaryk University in the Czech Republic. Her research focuses on authentication from the usable security point of view, particularly user perception of two-factor authentication.

**Milan Brož** is a Program Manager for Red Hat Research in Europe, maintainer of Linux® disk encryption, and researcher in the area of storage security. With over twenty years of industrial experience as a software engineer, he focuses mainly on storage and security systems design.





## What Is Usable Security?

Usable security is commonly understood as an interdisciplinary approach in which the ultimate goal is to balance security and usability. The problem of balancing usability and security was first considered by two groups of researchers: Adams and Sasse<sup>1</sup> and Whitten and Tygar.<sup>2</sup> They pointed out that the reason users do not behave securely in the IT domain is a lack of usability in security systems. In the previous century, users were simply assumed to be careless about IT security, because security is not usually their primary goal. Then Adams and

Sasse demonstrated the role of usability in the context of password policies. Near the same time, Whitten and Tygar revealed the importance of usability when researching how cryptography novices encrypt an email. That started the era of “Why can’t Johnny encrypt” studies, where even after many years there is still room for improvement. Their work shows that it is necessary to think about users when designing for IT security. For more details, see Martin Ukrop’s article [“Don’t blame the developers: making security usable for IT professionals”](#) in *RHRQ* 2:2.

Do independent developers use strong authentication? What kind of strong authentication is considered user friendly, even in a situation when developers need to use recovery options? To start answering these questions, we asked developers at a large open source software company.

### WHAT DO USERS ACTUALLY USE?

In our survey, participants were asked about their GitHub account usage, which type of projects they have maintained or contributed to in the last year, and whether they use any of the two-factor and fallback authentication methods offered by GitHub. The majority of them responded that they use two-factor authentication (and then have enabled fallback methods), which is a very positive finding. The authentication methods most often used as second factors were hardware and software tokens, with a larger representation of the software token. This could be influenced by the fact that our survey participants have to use

hardware or software tokens for their internal single sign-on systems, so all of them have experience with at least one type of token. That experience could also influence their perception and willingness to use two-factor authentication in GitHub. Independent contributors who are not forced to use two-factor authentication for any other service could behave differently.

Among the developers we surveyed, some teams could use a GitHub subscription and have an authentication defined in team policies. In the survey, having a policy that specifies what GitHub authentication they should use was reported by 26 participants, while 37 reported not having such a policy, and 39 did not know. Since no survey participants were bound by an official company-wide policy regarding

GitHub authentication, some policies are applicable only to some teams. A disturbing finding is that more than one-third of participants were not sure if there were any rules applicable for them. Both maintainers and contributors of open source and company projects were present in this group, as in the other two.

Another issue to keep in mind is that developers who responded to our questionnaire classified themselves as IT security savvy. It is possible that the questionnaire attracted responses from participants who consider security an important topic, so they are aware that it is important to constantly improve the security of open source repositories. If we have only this subsample of more aware users, the results for the whole population may not be as positive as the responses suggest, because

<sup>1</sup> Anne Adams and Angela Sasse, “Users are not the enemy,” *Communications of the ACM* 42 (1999): 40–46. Available from doi:10.1145/322796.322806.

<sup>2</sup> Alma Whitten and Doug Tygar, “Why Johnny can’t encrypt: a usability evaluation of PGP 5.0,” *Proceedings of the 8th Conference on USENIX Security Symposium* 8 (1999): 169–83.

two-factor authentication would be used only by IT security savvy users.

### **NO FACEBOOK FOR US!**

The participants also perceived almost all offered authentication methods as (rather) positive in terms of usability and security. The only method perceived as not usable and not secure was using a Facebook account as a recovery option. For this method, we got the lowest number of answers; participants very often skipped this question. To be able

### **Why GitHub?**

We chose GitHub because it is one of the most used development platforms for open source. When there was a migration from GitHub to GitLab, we modified our questionnaire for GitLab users so participants who did not have a GitHub account could participate as well. However, we received only 3 answers from GitLab users, so we included only GitHub.

to use this method, users must have an account on a specific social network. For recovery codes and fallback SMS codes, users do not need anything in addition to what they already have. In addition, Facebook is a third party for which users have to ensure security; if they use it only for GitHub, it could be perceived as burdensome. Nevertheless, for the users who already have a Facebook account, it could be perceived as a good

fallback choice available just for a few clicks (which is also evident from the fact that few users rated it as very positive in terms of usability and security).

However, participants rated not only usability of a social network account as rather low, but security as well. These are not surprising results, because the security is outsourced to another company. And not just any company: Facebook has been in the news because of data breaches exposing users. In another user study about digital identity providers for online banking, our participants were not able to agree on any appropriate provider, but they were uniform in not wanting to use Facebook. As one participant said, "I would not trust Facebook; that is probably quite logical." Respondents also mentioned that they interconnect their Facebook account only with not-so-important applications, such as applications monitoring exercise or e-shops. Facebook is a private-sector, international entity, which could be positive, because it works everywhere, but that also makes it hard to regulate and control the data processing.

Further, when a user loses access to the account, access to interconnected accounts could also be lost. That raises another question: does deployment of a non-preferred authentication method influence user perception of the overall security of the secured website?

### **DEVELOPER PARTICIPATION**

To gather the data, we asked developers for participation primarily via internal mailing lists. We started distribution via mailing lists intended for non-official

communication inside the company and the company newsletter. However, the response rate was very low: we got only 34 complete answers from 136 accesses to the questionnaire after one week. When the research was advertised via the official mailing lists from the research manager's office, we got an additional 116 accesses with 55 complete responses in the following two weeks. The data were collected from November 9, 2020, until the end of the month.

Around 10,000 employees were recipients of the aforementioned mailings. Most of them met the survey entry conditions (having a GitHub account), and it was possible to fill it out during working hours. Even without looking at the results, we should think about possible interpretations of the response rate. The first possible explanation is that recipients simply overlooked this email in the number of incoming mails. This is understandable but problematic, because it is difficult to draw generalizable conclusions from the small number of answers.

Another possible explanation has more severe consequences. The low number of clicks on the link could indicate that the topic itself was not interesting enough to developers. If this were the real reason for the low response rate, that may indicate how users perceive the security of their GitHub accounts. Security emphasis is often on code integrity and related commit authorization, which is also an important topic. Fortunately, it has plenty of attention. However, user authentication



is related to code integrity as well. When someone commits a code under a stolen identity, it could also have serious consequences.

## IS USABILITY FOR IT PROFESSIONALS NECESSARY?

Most studies focus on regular end user authentication, where the problems are different than those of IT professionals. Regular end users often struggle with how to use a token at all, or sometimes have trouble installing a software token into their smartphone, which may prevent them from using it. By contrast, some may consider IT developers or other tech professionals to be more skillful and more aware of IT security than regular end users, so there is no need to focus on their understanding of strong authentication. However, one of the top three endpoint security risks in 2019 was a “compromise of privileged users or system administrators.”<sup>3</sup>

As discussed above, compromising the account of an independent developer can severely damage a project or company. At GitHub, breached credentials were used to attack the repositories,<sup>4</sup> and a Microsoft GitHub account was hacked,<sup>5</sup> proving that the risk is serious. We have to keep in mind that not every IT professional is trained in IT security. And even if there is not a barrier such as, for example, using


---

For open source software companies, improving the security of independent developers helps to prevent issues in the whole supply chain later. It is also quite crucial for enhancing the security of the entire open source ecosystem.

---

cryptographic libraries during coding, we cannot assume that IT professionals use the most secure authentication methods simply because they probably know how. In fact, even the studies of IT security experts demonstrate that while IT security professionals follow security recommendations more often than regular end users, usability is an important factor for them as well.

## BUILDING SECURE PRODUCTS

Contributions from many volunteers or external developers are among the most important factors for the success of projects based on open source. For open source software companies, improving the security of independent developers helps to prevent issues in the whole supply chain later. It is also quite crucial for enhancing the security of the entire open source ecosystem. We hope that our research helps to highlight the importance of secure authentication in public repositories for our future product security. More detailed results will be part of the academic publication. When the results are published, the link will be available at Red Hat Research on the [project page](#). 

## ACKNOWLEDGEMENTS

*Thanks are due to Vashek Matyáš for supervision, Red Hat for support, and last but not least all those who participated in the survey and/or shared their comments via e-mails with us.*

---

<sup>3</sup>Jai Vijayan, “Assessing cybersecurity risk in today’s enterprises,” Dark Reading Reports, October 2019. [https://www.anomali.com/files/Dark\\_Reading\\_Assessing\\_Cybersecurity\\_Risk\\_in\\_Todays\\_Enterprises\\_Research\\_Report\\_Anomali.pdf](https://www.anomali.com/files/Dark_Reading_Assessing_Cybersecurity_Risk_in_Todays_Enterprises_Research_Report_Anomali.pdf)

<sup>4</sup>Michael Mimoso, “Breached credentials used to access GitHub repositories,” Threatpost, June 17, 2016. <https://threatpost.com/breached-credentials-used-to-access-github-repositories/118746/>

<sup>5</sup>Elizabeth Montalbano, “Report: Microsoft’s GitHub account gets hacked,” Threatpost, May 8, 2020. <https://threatpost.com/report-microsofts-github-account-gets-hacked/155587/>

## Interview

# The right idea at the right time: networking researchers use open source for real-world results

by Toke Høiland-Jørgensen

**W**e invited Red Hat Principal Kernel Engineer Toke Høiland-Jørgensen to interview **Anna Brunström**, currently a Full Professor and Research Manager for the Distributed Systems and Communications Research Group at Karlstad University, Sweden. Prof. Brunström has a background in distributed systems, but her main area of work over the last years has been in computer networking. Their wide-ranging conversation covers programmable networking, open data, diversity in IT fields, and more.



## About the Author

**Toke Høiland-Jørgensen** is a Principal Kernel Engineer working on networking and BPF. He holds a PhD on the topic of network performance and bufferbloat.

**Toke Høiland-Jørgensen:** Let's start by talking about how open source and research interact. In the academic community there has been a trend toward open access and reproducibility. How do you view the relationship between open source and the academic community?

**Anna Brunström:** I think open source has a very important role here. If we talk about systems research or networking research, the availability of Linux® and FreeBSD and the possibility to implement and test things in a real network stack has had a large impact. If you try to implement a new feature in a protocol or some other mechanism, it will be extremely difficult to build the entire network stack just to experiment. But when you have the open source available, you can implement it as a PhD student or researcher in a real system and run experiments in simulated and real networks. The networking research community has used this and gotten valuable results.

There is also the possibility of having a real-world impact, right? When you develop open source

solutions for things you research or invent, you can put them to practical use. This is very important for academia, to not just publish papers but also have an impact on industry and on society at large.

**Toke Høiland-Jørgensen:** I agree. At the same time, having moved over to the open source development side, I've also run into this perception that academics just write and publish papers and don't get the code ready or get it upstream. Have you seen this?

**Anna Brunström:** You see both sides of it, right? This depends a lot on personal interest or the interest of a group, because going from a research prototype to something upstreamed and integrated in the Linux kernel is a lot of extra work. Depending on when the PhD thesis needs to be done or the interests of the PhD student, this can vary a lot. But a lot of researchers do upstream results; there are research groups that contribute a lot to the implementations all the way up. But it's also very dependent on the interest from the open source side for the solution.





*"My perception is that the open source community is quite willing to help. If they see a useful solution from academia, they are willing to offer their advice on how to get it integrated." —Anna Brunström*

**Toke Høiland-Jørgensen:** We've just had a really infamous incident with graduate students from the University of Minnesota who intentionally introduced faulty code to the Linux kernel as part of their research projects. Is that a risk of this kind of interaction?

**Anna Brunström:** That was an unusual incident—that's not how the interaction normally works. But understanding the upstream process is definitely a hurdle to get over. This is why contact between academia and industry is very

important, to support that process. My perception is that the open source community is quite willing to help. If they see a useful solution from academia, they are willing to offer their advice on how to get it integrated. One of the issues is finding the right person to connect to or the right channels for getting this interaction.

**Toke Høiland-Jørgensen:** What struck me as a PhD student was how much effort goes into finding someone who can give you feedback. You're working

with something so specialized that there are very few people who can give you qualified feedback. The open source world is a lot the same. You can submit your proposal to experts and they will give you minutely detailed feedback. It can be an intimidating experience, but it's also liberating and incredible to get this level of expert interaction.

**Anna Brunström:** That is a good point. A lot of the research process is about finding people who have competence and interest in the same topics. When

The trend now towards more reproducible research and making the artifacts produced available as open source and open data is also extremely positive.

you submit for open source, you have these experts, but the experts are also quite busy. So it also depends on whether what you're proposing seems like an interesting contribution at that point in time. Maybe it is a good idea, but it's not useful right now, or it needs to be reshaped quite a bit. With all these things there's a timing issue for when ideas are proposed and how they fit into what the rest of the open source community is working on and prioritizing at the time.

Also, a lot of the functionality you have available in the Linux kernel is not used a lot. For instance, you have a lot of different congestion control algorithms. If you look at this from a researcher's perspective, it's a random selection. Why do we have this particular set of algorithms available? It has to do with the interests of the people who worked on those algorithms, the timing of when these algorithms were proposed, and what the interests of the rest of the community were. Also, many other algorithms are implemented, but they aren't implemented in the mainline. Ideas take a long time to mature, and you have ideas that build on each other. Not everything has to be integrated in the mainline kernel.

The trend for reproducibility and open data is important in this context as well. Maybe you have some iteration on the research side where you build on things, and then just a few of them get upstreamed. That whole chain of things has contributed to the particular algorithms or solutions that actually get upstreamed.

If all research ideas went up into the mainline kernel, that would be an indication that there is not enough exploration and innovation on the academic side. Not everything should be ready for production or working as well as you expect, because then it's not research.

**Toke Høiland-Jørgensen:** Open source is in itself really liberating for a PhD student. You can poke at the internals, learn how stuff works, and improve upon it. Even if that doesn't end up being part of a production system, it's still an important property of open source software.

**Anna Brunström:** Absolutely. We've had both Linux and FreeBSD available as open source for a long time, which has had a huge impact on networking research. You can also see that the bar is quite high. If you want to publish research in this area in top venues, you have to have an implementation in a real system, in some form.

The trend now towards more reproducible research and making the artifacts produced available as open source and open data is also extremely positive.

**Toke Høiland-Jørgensen:** I remember with some of my first papers, I had to struggle to get permission or funding to publish them as open access. During the time between then and the last paper I published, this changed a lot.

**Anna Brunström:** Definitely. It's also changed a lot from the funding agencies.



Now if you get the funding from the European Union or from National Funding in Sweden, for instance, the funders will require that you make your results available and push for open access. There's also a big change in program committees for conferences in the networking area. It's clearly valued if the paper makes the code or data available. It's now commonly considered in the review process when evaluating papers. Definitely a big difference over the last few years, and a very good development.

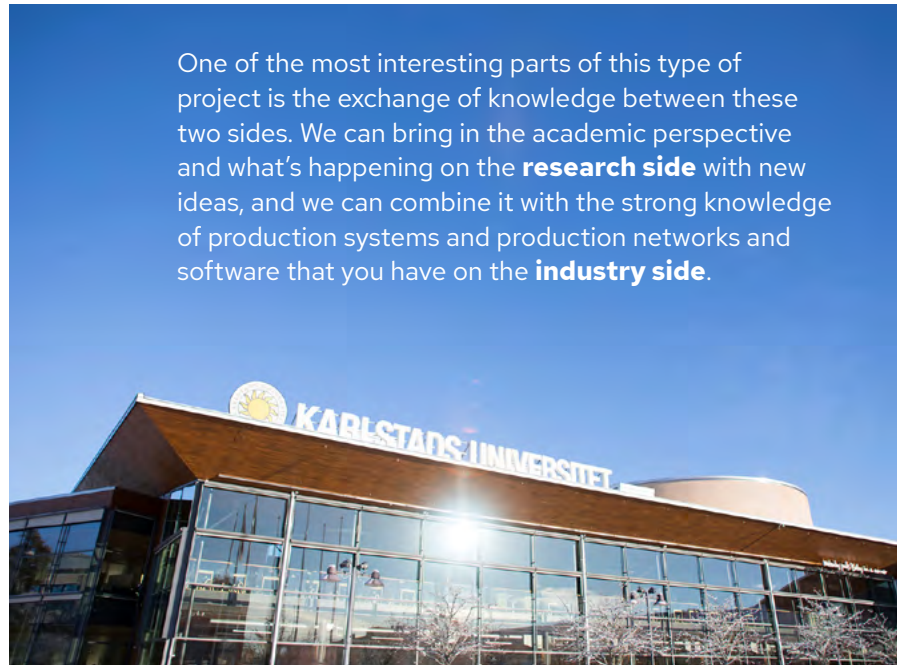
**Toke Høiland-Jørgensen:** Let's segue into the research collaboration set up now with Karlstad University and Red Hat. Why did Karlstad University decide to engage in this formalized research cooperation with Red Hat, and what's the most interesting part of the project?

**Anna Brunström:** First of all, it's an interesting topic. Programmable networking is a big trend now. It creates a lot of possibilities and flexibility, but it's not an easy environment to develop in. One of the most interesting parts of this type of project is the exchange of knowledge between these two sides. We can bring in the academic perspective and what's happening on the research side with new ideas, and we can combine it with the strong knowledge of production systems and production networks and software that you have on the industry side. One of the most interesting benefits of the project is the possibility for our PhD students to get detailed feedback on the code level.

**Toke Høiland-Jørgensen:** What is programmable networking, and why does it interest you?

**Anna Brunström:** Programmable networking means that you can manage and control the network through an open API. So you can modify how the network behaves by reprogramming

One of the most interesting parts of this type of project is the exchange of knowledge between these two sides. We can bring in the academic perspective and what's happening on the **research side** with new ideas, and we can combine it with the strong knowledge of production systems and production networks and software that you have on the **industry side**.



it, in a much more flexible way than you could before. One important factor is that the control has shifted away from the vendors of networking gear. Before, if you wanted some new functionality, it was very hard to achieve, and it was a very long-term process. But when these devices become programmable, the network owner and network operator can develop software that can then affect how the network operates. This shift of control also opens up the network for innovation and exploring new functionality.

**Toke Høiland-Jørgensen:** I always viewed programmable networking as the logical evolution of software-defined networking (SDN). Would you agree?

**Anna Brunström:** Yes. I would say that SDN is programmable networking. If we go back in history, we have active networking, which was a research area in the late 1990s. I think that was the first trend where researchers tried to dynamically

---

Programmable networking means that you can manage and control the network through an open API. So you can modify how the network behaves by reprogramming it, in a much more flexible way than you could before.

---



update or program the functionality of the network. And then we had the early research on separating the control plane and the user plane, which is one of the foundations of SDN.

Then came OpenFlow, which is the most well-known protocol in SDN. I remember the SIGCOMM conference, I think it was in 2009 in Barcelona. They had all the demos on OpenFlow and all the things you could do with it—it took over almost one of the rooms. It came out of Stanford, so they had a lot of demos at the conference and we were all thinking that all this was a pretty cool thing. Then there was the idea of separating the user plane and control plane, then being able to update what

happens in the data plane through this decentralized control. That is very much programmable networking, I would say. The next step is that the data plane itself has become more programmable.

So if you take OpenFlow, you could program the network, but the functionality that you could put in was still fairly restrictive. You had a limited number of what you could match packets on and what actions you could apply. With the programmable data planes you have the possibility to implement your own matching rules and the programmable match action tables. You have an increasing level of control, but you still have the separation between the control plane

and the data plane, and you update the network through the control plane.

**Toke Høiland-Jørgensen:**

One interesting aspect of this is the convergence between the software world and the traditional networking world, which has been based on hardware vendors for a long time. You had these big iron boxes that you forklifted into your datacenter, and they would process billions and billions of packets per second in a single refrigerator-sized device, versus software based on off-the-shelf hardware that you can just buy in bulk and scale.

Coming from the hardware side, there has been this idea that a networking protocol or packet processing is static. But programmable networking is really about bringing this fungibility of software—the ability to change everything—into these big iron boxes as well, right?

**Anna Brunström:** Yes. And this was also the driver, that it was very hard to manage networks. So, it's the same trend, and you continue along this scale of things getting more and more flexible. And along with this you have the trend towards virtualization, which fits very well with programmable networking. You can use programmable networking to implement virtualization, but it's also a concept of its own.

There is also the trend towards open hardware and disaggregation of hardware, whereas before you had a



lot of functionality put together in one box. Breaking the box down into smaller pieces opens them up for innovation or evolution and access to other players. If you have to bundle everything into a big box with all the hardware for all this functionality and all the software and the logic for it as well, then contributing to that market is much different from what we have now, when you can run on disaggregated hardware and general purpose hardware, and use open source to manage all of these components.

**Toke Høiland-Jørgensen:** And it's the same code that will run on the datacenter and in the small devices you have at home or in your pocket. You can contribute to all that, even though you don't have a datacenter yourself. You get the benefit from all of it everywhere because of this ability of software to transcend barriers.

**Anna Brunström:** True. You also have open interfaces and open source that allow you to adapt it and modify it. You open up that possibility to a large number of people when you have open source software to start from and you have an open API and it's available. If that software is produced and then put in a box, no one can affect that. If you want something done to it, you have to hope that the vendors can support it. It's also good from a hardware perspective because you can optimize the hardware to support the key principles for programmability and flexible functions, rather than trying to select which protocols should be supported in these boxes.

**Toke Høiland-Jørgensen:** That was one of the surprising results to me. Some of the prototypes of these completely programmable data planes had better performance than the specialized hardware. It turns out that if you come up with the right primitives, you can optimize the hardware to execute those with performance that more than matches the specialized things you had before.

**Anna Brunström:** Yes. If you have to implement everything in hardware, you have to put a lot of work into supporting all these various things. Whereas if you can focus on supporting a good set of base primitives, then you also have the possibility to do that better.

**Toke Høiland-Jørgensen:** It's also cyclic to a certain extent, right? It cycles back and forth between general purpose processing and specialized hardware offloads.

**Anna Brunström:** With programmable networking, they get much more closely integrated, or the boundary between them is more flexible. There is a lot of research now that experiments with what functions you put in hardware, what functions you put in software. You have this continuum of different hardware and software solutions, and you can move the functionality in a flexible way between these different components. This also depends on your problem and how much that requires in terms of pushing packets or in terms of memory, for example.

**Toke Høiland-Jørgensen:** We are also moving from looking at specific devices connected to the network as a programmable entity that you can change the behavior of to looking at a whole network as one big programmable entity with SDN as the centralized view of the control plane. Everything becomes programmable. Where does this leave protocols and standardization bodies like the Internet Engineering Task Force (IETF)?

**Anna Brunström:** You still need the protocols, right? You can implement the protocol in different places, and you may implement part of the protocol in a programmable switch to speed things up, and you get feedback or you distribute the protocol in different entities. For some things you still need interoperability between different components. So I wouldn't say that the need for protocols disappears because of this. You still need to be able to interact. Also, protocols and standards are important in this domain because there's also an open interface between different entities in the network. And you have a lot of new protocols or interfaces connected to this that also need to be standardized.

**Toke Høiland-Jørgensen:** Part of the standardization process is needed and part of this is not, right? There's a whole lot more review, and people looking at things with different perspectives, but part of it is just an old slow process that needs to change.

**Anna Brunström:** Yes. But the processes can also interact in some

sense. If you go back to the beginning of IETF standardization, it was very much based on running code. That changed over some period, and some of it has not been as much based on running code. But because of things being more open source based, we're moving back. In the ideal world these two things interact, right? If we think about the QUIC protocol, for instance, that was standardized as a new transport protocol. That was developed hand-in-hand with the code that implements that protocol. And there were somewhere between ten and fifteen independent implementations of the protocol that were being developed together with the standard, and many of them open source.

**Toke Høiland-Jørgensen:** Yes. I think the standardization of the QUIC protocol is the best example of where this evolution is working, and where the standardization process interacts with the open source community in a good way. They interact with each other, and hopefully that will improve both.

**Anna Brunström:** When networking started, it was based more on open source. It came more from the academic community. Then it was dominated by industry and by closed source and vendors that control different network equipment over a period. With the networks opening up again, we can see that you have a closer interaction between the code development and the standardization.

QUIC is a very good example. And the way that IETF is working, at least in

the transport area, it's moving more towards open source-like development. Think about how documents are being developed now. We are using GitHub, and in the same way you develop open source code, you develop the standards. Lots of different people can contribute to different pieces of the standard, based on their experience. That reminds me more of how open source code is developed now than it did ten years ago.

---

With the networks opening up again, we can see that you have a closer interaction between the code development and the standardization.

---

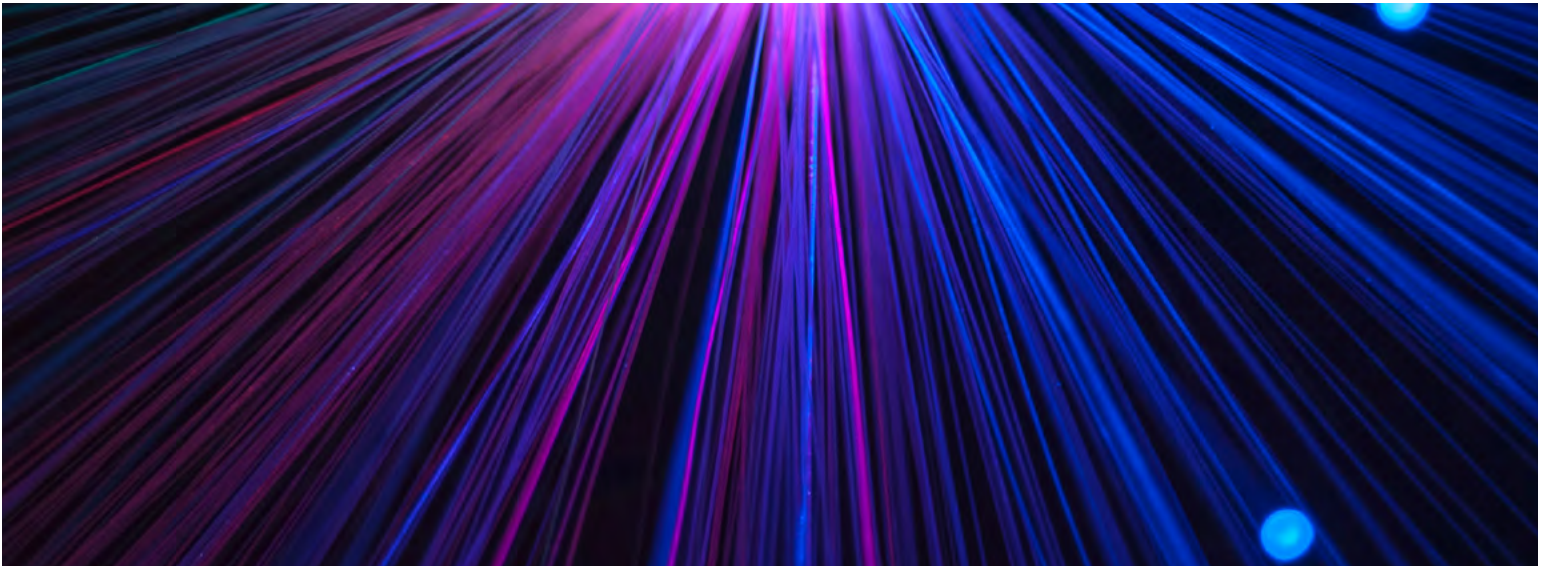
**Toke Høiland-Jørgensen:** Other than what we've already discussed, what would be some of the interesting trends in your field?

**Anna Brunström:** One thing we have not talked about, but that ties very closely into programmable networking, is how this now moves into other domains and into the cellular network. Looking at 5G and beyond 5G networks will be very important for the future, with all the connected devices and the communication going over these types of technologies. This trend of programmable networking and disaggregation is moving into the cellular world.

They are both working with disaggregation and open interfaces, in this O-RAN initiative, for instance. That's quite interesting from a research perspective, because it means you can contribute in that domain as a researcher. Another thing that ties into what we have talked about is the entry of AI into networking. You have the idea of the closed loop control, where if the network is programmable end to end and top to bottom, and you can get a lot of information out of the network, how do you then automate management of the network and optimize it?

The third thing is this general development with digitalization and how networking now is a component of almost any engineering field or any solution in society. There are a lot of interesting opportunities to collaborate with researchers from other fields in different applications. We have always worked a lot with people from networking or companies related to networking. But now we also see collaborations with companies from different domains that see networking and the data that is produced and data management as important tools in their development.

**Toke Høiland-Jørgensen:** One of the fascinating things about working with networking and the internet has been the sense of the whole world being within reach. On the one hand it's really scary, because suddenly everything is exposed to the internet. On the other hand, there's also this convergence of all of human society through this giant web of communication.



**Anna Brunström:** Yes. If you're a student in networking or computer science today, you can use your technical interest and your technical knowledge to work in almost any area and make a big impact on how things develop in society. So it's certainly an exciting time for researchers in our domain, and we're starting to see this in student recruitment.

**Toke Høiland-Jørgensen:** Do you also see this reflected in a larger diversity of backgrounds in the student body and in research in general?

**Anna Brunström:** That is still a challenge. I've seen some studies on younger ages suggesting that attitudes are changing in secondary school with regard to which fields kids can imagine working in. More girls, for instance, are showing interest in the field. I couldn't say that we see it in the students we have in our courses today. I certainly think there


is a potential for change, because more and more people realize what impact you can have through networking and through computer science knowledge, and how many different things you can achieve with a career in that area.

I certainly have good hope. I also see that the IT industry is very aware of the need to broaden the talent pool. We have good collaborations from our university and with industrial clusters in the region to try to attract a broader group of talent to the field.

**Toke Høiland-Jørgensen:** Opening up the field to a diversity of opinions is important because technology runs everything now, but it's also important for the quality of the technology. There are perspectives and use cases that are just not considered by the homogenous population in technical fields. But I'm cautiously optimistic that this is finally moving in the right direction.

**Anna Brunström:** This has been known for a long time, right? But progress has been slow. With some of the trends that we see today, I'm hopeful that it will speed up. And there is greater understanding of the importance of these technologies. A lot of fields that have not thought much about networking and data management in the past are now thinking about these areas to reform their businesses and production environments. There are an endless number of applications for these technologies.

**Toke Høiland-Jørgensen:** Let's hope you are right that it will speed up. And I certainly agree about the endless number of applications! This has been a fascinating conversation, thank you for taking the time to chat with me.

**Anna Brunström:** You're welcome, it was my pleasure! 



## Feature

## Faster hardware through software

Researchers have tested several techniques for using software to get the most out of hardware. Find out about three promising projects that indicate the direction of this quickly changing field.

by Gordon Haff



### About the Author

**Gordon Haff**

is Technology Evangelist at Red Hat, where he works on emerging technology product strategy, writes about tech trends and their business impact, and is a frequent speaker at customer and industry events. His books include *How Open Source Ate Software*, and his podcast, in which he interviews industry experts, is *Innovate @ Open*.

It used to be simple to make computer workloads run faster. Wait eighteen months or so for more transistors consuming the same amount of power, maybe some tweaks in the fabrication material, some thoughtful balancing of frequency, core counts, interconnects, and cache sizes, and you were good to go. I'm trivializing a lot of difficult engineering work on both the hardware and software side to be sure. But the essential point is that the processor and other components largely got faster in a way that was mostly transparent to software.

With the slowdown of increasing transistor density provided by Moore's Law and the end of the static power density offered by Dennard Scaling, this free (or at least relatively cheap) lunch is ending. Software needs to take advantage of any useful clues the hardware provides to help it run more efficiently, and the hardware itself needs to more intelligently manage its transistor budget.

The horizontal stack described by former Intel CEO Andy Grove, in which the processor and operating system were largely isolated from each other, is giving way to a world in which hardware and software are increasingly co-designed. In this article, I take a look at ongoing

research that starts to break down the hard abstractions between software and hardware.

### BREAKING DOWN ABSTRACTIONS

Can adding a single mechanism to Linux® enable huge performance gains? The Symbiote project led by Tommy Unger, a PhD student at Boston University (BU), enables an otherwise entirely standard Linux process to request *elevation*, that is, to run with full supervisor privileges. Elevation removes the hardware-enforced boundary between application and operating system code, allowing for aggressive co-optimization where desired. Importantly, the optimization methodology can be incremental. Before modification, Symbiotes are indistinguishable from Linux processes and are fully compatible with all the standard Linux interfaces. During optimization, programmers can settle on hybrids that partially utilize the Linux kernel, but also may specialize hot paths like system calls and interrupt paths. In the extreme, a Symbiote can fully take over the system paths and leave Linux behind—in this sense you might think of Linux acting as a bootloader for a library operating system.

Symbiotes completely relax the application-application and application-kernel protection domain. They can run virtualized or bare metal.

Unlike unikernels and library operating systems, Symbiote does not prescribe a fundamental change to the familiar virtual address space layout of a process and the Linux operating system. A Symbiote can run alongside other standard Linux processes, and it can `fork()`. One guiding principle for Symbiote design is keeping changes within the application executable (as opposed to making changes to the Linux kernel) whenever possible. For example, if a programmer wants to specialize the page fault handler, they can `memcpy()` the Interrupt Descriptor Table (IDT) into the application address space, modify it there to point to a user-defined handler, and swing the Interrupt Descriptor Table Register to point to this new IDT. We target lines of code changes to the Linux kernel to be in the hundreds.

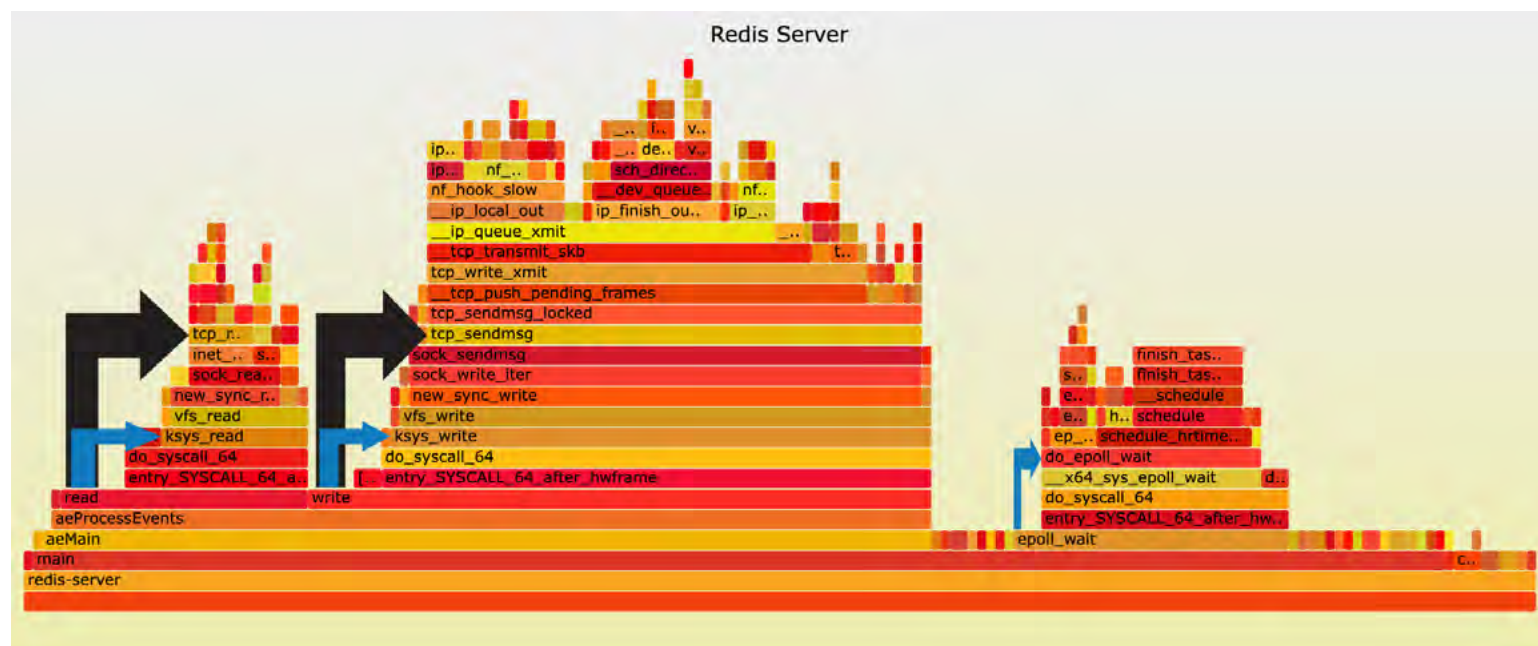
**Figure 1.** Redis server

Symbiotes allow for fast incremental optimization. Consider optimizing the Redis server. Running a profiler (see **Figure 1**) shows significant time is spent in the `write()` syscall. Elevating the process to run as a Symbiote allows for substituting `syscalls` for low latency (and low processor perturbation) `call` instructions. Further, removal of the protection boundary allows the application to call directly into kernel internals. It is easy to prototype application-specific specialization by calling progressively deeper into kernel paths. On Redis' use of the `write()` syscall, for example, the programmer can call directly into `ksys_write()`, skipping the syscall handler. Or they could push further into `new_sync_write()`, skipping the virtual dispatch in the VFS layer. Making aggressive use of application-specific knowledge, they might call directly from the Redis

server app into `tcp_sendmsg()`, skipping resolution of socket and protocol layers. This procedure allows for the rapid construction of highly optimized, one-off executables, and this effort can be partially amortized by building a shared set of optimization tools in a Symbiote library.

A programmer may choose to have their Symbiote play nicely with the underlying kernel and other processes, but it can also be used to strip away (or modify) Linux policies.

A process could take direct control of the interrupt and syscall paths (vectoring directly off a network interrupt to a handler in the app's address space); modify network protocols; change scheduling, preemption, and power policies; modify kernel data structures; or allow for direct



application control of devices. On the flip side, traditional kernel paths can be pulled into user space and extended with arbitrary code: consider that this allows for compile time optimization across the application-kernel boundary, or the replacement of a heuristic like a high-water mark with a neural network defined in a high-level programming language.

#### **EVEN A LOWLY NETWORK CARD NEEDS HELP**

Intelligent performance tuning can even bring benefit to a device as seemingly simple as a Network Interface Card (NIC). That's because even a "simple" NIC, such as the Intel X520 10 GbE, is complex, with hardware registers that control every aspect of the NIC's operation, from device initialization to dynamic runtime configuration. The Intel X520 datasheet documents over 5,600 registers—far more than can be tuned using trial and error.

BU PhD student Han Dong's project aimed to identify, using targeted benchmarks, those application characteristics that would illuminate mappings between hardware register values and their corresponding performance impact. These mappings were then used to frame the NIC configuration as a set of learning problems such that an automated system could recommend hardware settings corresponding to each network application. This, in turn, allows for new dynamic device driver policies that better attune dynamic hardware configuration to application runtime behavior.

Dong specifically studied different operating packet processing structures and their implications for the energy management of network-oriented services. He conducted a detailed operating system-centric study of performance-energy trade-offs within this space. Two hardware "knobs" on modern Intel servers control these trade-offs: specifically, 1) Dynamic Voltage Frequency Scaling (DVFS) that throttles processor frequency to save energy and 2) the interrupt delay register on modern NICs to control network interrupt firing rate. Both of these hardware controls exist in the general context of methods that can throttle network application processing.

To gain further insights into the effects of how slowing down can impact these trade-offs, he studied manually controlling these two knobs under different traffic loads of four network applications across two different operating system structures: a general purpose Linux kernel and a specialized packet processing operating system (akin to a unikernel or an application built using network acceleration libraries such as the Dataplane Development Kit).

Among the findings were that a pure network poll could result in better performance and energy consumption than being interrupt driven; the entire workload finished quickly without using sleep state idle modes. A specialized operating system path length can also reduce time spent in processing by improving application

code instructions-per-cycle (IPC), even with application-intensive workloads. This creates additional headroom to use slowing down processor and network interrupts to further reduce energy use. This specialized stack can more aggressively slow down processing to save energy by up to 76 percent over a general purpose operating system, with no noticeable impact on overall performance.

#### **OPTIMIZING DATA ANALYTICS CLUSTERS**

Another performance optimization project comes out of BU, Northeastern University, and the Mass Open Cloud. The Kariz cache prefetching and management project focuses on the performance of data analytics clusters that usually segregate data into large multi-tenant data lakes, which are often relatively low performance relative to local storage.

Like some of the other research described, Kariz takes advantage of information that the system is already surfacing. In this case, it's I/O access information that already exists for use in scheduling and coordinating distributed workers. Spark-SQL, Hive, and PIG all collect this information in the form of a dependency Directed Acyclic Graph (DAG) identifying inputs and outputs for each individual computation. Given future access information (e.g. job DAGs), Kariz determines which datasets to make available in cache by either prefetching or retention. Kariz determines which data to prefetch or evict, and when to do so.



Kariz also revisits some common assumptions. For example, it asserts that hit ratio and performance are not necessarily directly related, input files don't need to be cached in their entirety, and the limits of back-end storage bandwidth are a relevant metric.

## **MAKING MEMORY LESS OF A BOTTLENECK**

Other research at the Red Hat Collaboratory at Boston University has focused on kernel techniques that optimize memory bandwidth while keeping latency predictable.

Recently, memory density has increased vastly, thanks to Non-Volatile Dual In-Line Memory Modules (NVDIMMs). But memory bandwidth has long been one of the most limiting factors on overall system performance; CPUs can execute dozens or even hundreds of instructions in the time it takes to access memory just one time. Therefore the larger but slower NVDIMMs coexist with the faster DIMMs in a system, where they often serve as a main memory cache. The caching scheme and logic used between the DRAM and NVDIMM memory require kernel knowledge to maximize memory bandwidth and minimize memory latency.

As a result, a project between BU and Red Hat is exploring how the Linux kernel can take advantage of the newest CPU hardware features and system memory topologies. NVDIMMs running in memory mode (they can

also operate as fast storage) is the near-term future of computers and must be optimized. Currently the Linux kernel has no way of evenly distributing the pages of NVDIMM memory throughout the DRAM cache, and this results in difficult-to-predict memory bandwidth and latency. A technique known as page coloring will be investigated and evaluated. A significant amount of work has been done for Non-Uniform Memory Access (NUMA) placement to reduce the number of remote (higher latency) memory accesses in more traditional CPU interactions with main memory, but NVDIMM optimizations will likely have to take different approaches.

The most recent result from this area of research is presented in [“E-WarP: A system-wide framework for memory bandwidth profiling and management”](#) by Renato Mancuso, Assistant Professor in the Department of Computer Science at BU; Parul Sohal, a PhD candidate at BU; and Red Hat Distinguished Engineer Uli Drepper.

It proposes an Envelope-aWare Predictive model, or E-WarP for short. It's a methodology and technological framework to:

- analyze the memory demand of applications following a profile-driven approach,
- make realistic predictions of the temporal behavior of workloads deployed on CPUs and accelerators, and

The caching scheme and logic used between the DRAM and NVDIMM memory require kernel knowledge to maximize memory bandwidth and minimize memory latency.



- perform saturation-aware system consolidation.

The goal is to provide the foundations for workload-aware analysis of real-time systems.

### THE ROAD AHEAD

Using advanced software techniques to get the most out of hardware, often dynamically, has been the focus of this article. However, as noted in the introduction, there's an increasingly symbiotic relationship between software and hardware, in that the software can do the best job when the hardware surfaces need information.

Processor hardware is also increasingly implementing dynamic optimizations on its own. Power management is the best-known (and most long-standing) example, but we see more far-reaching optimizations coming online in newer processors.


For example, Intel's 3rd Gen Intel Xeon Scalable processor (Ice Lake) has several features in this vein. Intel Speed Select Technology (SST) offers a suite of capabilities to allow users to reconfigure the processor dynamically, at runtime, to match the workload by tuning the frequency, core count, and thermals. Intel Resource Director Technology enables monitoring and control of shared resources to deliver better quality of service for applications, virtual machines (VMs), and containers.

With the evolution of a feature like SST from something that had to

be set in the BIOS at boot time to something that can be dynamically changed at runtime, we see just one way in which software and hardware can work together to maximize the efficiency of a running program.

### MORE TO COME

We'll continue to probe these developing areas of research and technology in future issues. Unikernels complement the operating system structure work described in this article. We plan to dive deeper into the processor optimizations I've touched on here. And we'll continue to keep tabs on the ongoing research around Field Programmable Gate Arrays (FPGA), flexible chips that can be programmed again and again with different code paths for different workloads. One ongoing project aims to use machine learning to control a newly customizable version of the GNU C Compiler (GCC) to automatically determine optimization pass ordering for FPGA targets specifically, and thereby improve performance as compared to existing proprietary C-to-FPGA methods.

Software and hardware architectures haven't been this interesting for a while. 

### ACKNOWLEDGEMENTS

*The author would like to thank Red Hat's Larry Woodman and Uli Drepper, BU Professor of Electrical and Computer Engineering Orran Krieger, and BU PhD students Tommy Unger, Ali Raza, Parul Sohal, and Han Dong for their assistance with this article.*

# “When one teaches, two learn”: making the most of technical research mentorship

Research mentorships are the basic building block of productive industry-university relationships. We asked four mentors from around the globe to tell us about the challenges, rewards, and strategies of serving as a mentor.

by *Matej Hrušovský and Lis Strenger*

**L**inking a student’s research goals with the experience of a Red Hat software engineer is at the crux of the Red Hat Research program. Even formal partnerships with universities boil down to the one-on-one relationship between student researcher and industry mentor.

To find out what is involved in being the non-academic mentor of a computer science researcher, *Red Hat Research Quarterly* interviewed a few Red Hat mentors to discuss the hows and whys of these connections. Our mentors (see bios in insets) range from very seasoned to very new, and they work with students at universities in the US, Israel, and the Czech Republic on an array of projects.

Learning more about the dynamics of mentoring research projects will help us better understand the benefits and areas for improvement in research mentorship programs. After all, these relationships are uncommon in both commercial and academic settings. But they do important work by bridging these two spheres, translating knowledge and experience into insights that

help student researchers’ work become more immediately relevant to applied computer science.

In the questions and answers that follow, we synthesized some of our findings while letting each mentor speak from their unique experience.

## HOW DO YOU PACKAGE YOUR ADVANCED TECHNICAL KNOWLEDGE SO THAT IT’S SHAREABLE WITH STUDENTS?

Mentors have a wealth of information to share, but they aren’t classroom teachers. Imparting what they know to their mentees is a matter of individual communication styles, learning styles, and coding styles that will likely differ in each mentor relationship. We asked mentors how they like to illustrate complex ideas for students. In some instances, merely simplifying the explanation of a concept is enough. Often, however, mentors find that showing rather than telling is a more effective way of helping a student learn a new idea.

Uli Drepper, for example, likes to write up code that implements a concept, then let the student have at it. Students can work with the code themselves





**Josh Salomon** is a senior principal software engineer at Red Hat working as an architect on Ceph storage systems. Based in Israel, Josh has worked on projects with professors and students at the Interdisciplinary Center (IDC) Herzliya.

until the concept is grasped to the point where they can apply it in their work. This is a very practical approach: it leads to results very quickly, something students appreciate because it gets their projects off the ground quickly too.

Josh Salomon prefers using the whiteboard to provide high-level diagrams that explain everything he needs to about a project. The whiteboard has long been one of the most natural ways for an engineer to explicate a concept. Unfortunately for recent mentorships, that option became a casualty of the pandemic. After COVID-19, he provided these explanations remotely, with a heavy reliance on documents and written explanations.

When a mentor has more actual teaching experience, their starting point might not be the subject matter, but the students themselves. After assessing students' technical knowledge, Viktor Malik shapes explanations to match. He believes strongly in showing everything through examples of code so that the explanations are not abstract. Instead, students can see how something works in practice and start applying it to their own work.

Except when they can't: sometimes complex concepts take repetition. When that happens, Viktor breaks things down into smaller steps and moves through them more slowly. "Students can become distracted by complexity," Viktor said. But once they are able to move through the problem independently, "I've been surprised at students' creativity and resourcefulness. Sometimes they come up with solutions I hadn't considered that ended up being much better than my approach would have been."

Even when allowing students to figure things out for themselves, there is a

role for mentorship. Marek Grác likes to create a context for students to puzzle through a problem, which is also a great way to test motivation—are students willing to put in the work it takes to grasp computing concepts? Marek provides consistent support along the way by providing pointers as needed. According to Marek, "The best kind of knowledge I can provide is not necessarily the details of a particular subject, but a larger understanding of how to propose a hypothesis and then how to support it."

With this approach, the mentorship is really based on dialogue: the mentor may provide suggestions, but conversations are mostly a back-and-forth sharing of ideas. That kind of collaborative mentorship can lead to unexpected results. For example, one of Marek's mentees was working on a thesis about an idea for which Marek was the original researcher. Rather than playing the expert, Marek sent the student off to do their own exploration, which led to the student discovering new issues that Marek had been unaware of. Success!

#### **HOW DO YOU KNOW WHEN TO STEP IN AND WHEN TO STEP BACK?**

Finding the sweet spot between direct knowledge transfer and independent discovery takes patience and a bit of self-control. The temptation to make students' lives easier by providing shortcuts or quick insights can be great, especially when a mentor is excited to help the student's

project come to life. The mentors we interviewed had clear ideas about when their assistance made sense.

Setting the design and technical parameters of a project is often done in a collaborative manner. A mentor has to realize that they might have their own vision for how a technical problem can be approached, but they need to allow the student to bring theirs to the table as well. Mentors preferred taking a hands-off approach to the students' work once the big picture and goals were clear. The most common reasons for offering more direct guidance were to help navigate complex or unfamiliar elements of the open source process and to help students focus on the practical, applied



---

When students are working on an open source project, they find that there is a big gap between what academia wants and what the open source community wants. The community wants something that works well, and in academia they want something that is explained well.

---

outcomes of their work. After all, students choose to bring academic work to the Red Hat Research program explicitly because of this practical aspect, with the hope of having their work become part of actual enterprise solutions via the open source channel.

Once mentors have helped identify which open source project best aligns with the student's work, they can help increase the likelihood that a student's work will be accepted by the project's maintainers. Some open source projects are quite complex, with multiple subsystems and layers. Students benefit from having someone to guide them through where to put the code, how to build it, and so forth. Josh, for example, is deeply knowledgeable about the Ceph project, but he was nevertheless grateful that one of the core Ceph developers

was very keen to assist his student and guide her through the contribution process. However, how she actually wrote the code and solved technical issues along the way was up to her.

It is the nature of academic research to drill into a particular aspect of a problem and thoroughly understand it, but the student's ultimate goal while working with Red Hat Research is to prove a hypothesis and finish the project. Marek finds himself reminding students of this goal as a way of bringing their focus back to the project at hand—and to the student's long-term academic and career goals. It might not be in the student's best interest to follow interesting research threads if their plan is to finish their degree and start working for a technology company. "When students are working on an open source project, they find that there is a big gap between what academia wants and what the open source community wants," Marek said. "The community wants something that works well, and in academia they want something that is explained well."

The student can then make an intentional choice whether to spend less time explaining a problem and



**Marek Grác** is a senior software engineer at Red Hat Czech who has lent support to thesis projects at Masaryk University in Brno at master and doctoral levels on topics including networks and AI/ML.

His solution is to have students take their more theoretical hypotheses and build an actual proof of concept as part of their projects' outcomes. For example, a current project around optimizing bytecode in Java will probably take years to be implemented to the standards of the Java community, but a student can create a working prototype to prove out the solution.

#### **WHAT ARE SOME OF THE COMMON CHALLENGES, AND HOW DO YOU HELP STUDENTS SURMOUNT THEM?**

Our mentors found that students' work habits are another area where they can offer some useful guidance. Naturally, some students tend to procrastinate, leave things to the last minute, and then try to finish everything in a short

**Uli Drepper** is a Red Hat Distinguished Engineer working on multiple projects as part of the Greater New England RIG. Uli currently advises on ten theses/dissertations at Boston-area universities, many of which are in the area of programmable hardware and the Linux kernel.

more time on the practical solution; in the end, the practical application will have more impact for more users once it has been contributed to an open source project. For example, one of Marek's students was working on reducing the energy consumption of a Wi-Fi chipset. His solution is now part of the Linux® kernel, which means it is implemented on millions of computers—indeed, every computer that is running Linux. From an academic point of view, this may have been an average thesis. But judged by its impact, it was an extraordinary technical contribution.

Marek also clarified that there doesn't always have to be a conflict between practical and academic accomplishment.

span of time. Another common problem is students who try to do everything, which often leads them to stray from their original project intent or goals.

For procrastinators, mentors have tried a variety of strategies. First, mentors can help by leading students toward a realistic timeline. Students often don't know how much time it will take to complete such a long-term task, and it frequently takes more than expected. Mentors can help set appropriate expectations from the start, for example, by requiring thesis projects to run at least two terms (one year). Second, task management and regular guidance are important, especially with students who tend to wait until the last minute, so they don't fall behind. Setting shorter-term milestones for the overall project and providing counseling as needed can make a big difference. Actually passing these milestones and getting the work done, however, is left purely to the student's own dedication.

Trying to do everything tends to be a problem of overeagerness, and it is a much more pleasant problem to have. "Many students like to build things from scratch," Uli observed, "which has obvious benefits, because it takes away constraints. But in real-world work settings, few people get to do that." Most work in IT consists of building on existing foundations. Uli added that a mentor should remind students that they need to ground themselves in doing a few things really well rather than many things badly, and go step by step to achieve that.



Marek usually discourages this type of student from being too creative once their goals have been set. When a project begins, there often seems to be lots of low-hanging fruit. The problem is that students often end up with a basket full of cherries only to remember that their original goal was to collect apples. Even very talented students can get distracted easily, and it is the mentor's role to keep them on track.

While mentors emphasized encouraging independent work, one of the most common issues they have is that students are afraid to ask questions, typically because they think they will look silly. A mentor can be a great informational resource, but to get the most from the mentorship a student needs to get over this fear. Instead of pushing students to ask, Marek turns the situation around. His students know a lot about certain subjects, so he simply asks them. That shows them that he, too, doesn't know everything, and that we are all learning.

## WHAT'S THE REWARD FOR YOU?

Red Hat's entire academic program is built on the work of volunteers who have to find time to carry out all these mentorship tasks. So why do they do it?

The first answer was unanimous: our mentors enjoy teaching. That is no surprise. "Helping people succeed comes with immense satisfaction," Viktor said, and most mentors would likely agree. Marek added that he enjoys staying in touch with new ideas, and that often means straying off the beaten path. Mentorship allows him to see what's

---

The problem is that students often end up with a basket full of cherries only to remember that their original goal was to collect apples. Even very talented students can get distracted easily, and it is the mentor's role to keep them on track.


---

happening in different disciplines and not concentrate too much on one thing. "The world needs not only experts, but also connectors," he said.

Uli noted another benefit that has long been a part of our academic programs: connecting demand with supply. The IT industry has no shortage of problems that need to be solved, features to be added, or tasks to be completed, and engineers often have to prioritize which are the most critical to the success of the project, product, or company. This leaves a lot of problems to be solved in the backlog.

The mentorship relationship creates a win-win situation: many items on

engineers' backlogs will never become a high industry priority, but they can be a basis for an interesting project and a learning experience for a student, especially under the proper guidance of a mentor who understands the problem. The work gets done, and the student comes out with valuable experience working on something real that may even get into the upstream. This important connection between university and real-world engineering would not be possible without a mentor's help.

To learn about the projects that are currently supported by Red Hat mentors, visit the [Research Directory](#). 

## About the Authors

### Matej Hrušovský

has been with Red Hat for more than eight years, six of which have been spent managing the university program in EMEA. Aside from attracting new talent mainly from universities and schools, the core of Matej's job is to find and put the right people from Red Hat and academia in the same room together.



### Lis Strenger is

Senior Manager, Emerging Technologies Marketing at Red Hat. With nine years at Red Hat, and over twenty-five years working in technology—a career which started as a college instructor of technical writing—she is captivated by the idea of learning through teaching.



## Feature

# BigDataStack delivers with contributions from industry and university partners

Data skipping and network performance improvement technologies prove their value in data-intensive applications.

*by IBM contributors Yosef Moatti, Oshrit Feder, Guy Khazma, Gal Lushi, and Paula Ta-Shma, and Red Hat contributors Luis Tomás Bolívar, Miki Kenneth, and Josh Salomon*

In 2018, fourteen industry and university partners launched a collaborative project designed to optimize big data analytics performance, efficiency, and scalability. Known as the [BigDataStack](#), the project, funded by the European Commission Horizon 2020 Work Programme, was featured for the first time in *Red Hat Research Quarterly* in the May 2019 issue. In January 2021, the BigDataStack consortium delivered their complete solution.

The project has more than met its objectives. The final review report opens by saying, “BigDataStack has delivered exceptional results with significant immediate or potential impact,” which testifies to the project’s quality. Indeed, many significant research efforts in this project led to state-of-the-art technologies, with no less than three chosen by the [EU Innovation Radar program](#), which aims at discovering cutting-edge EU-funded innovations. Since it would be impossible within the scope of this article to address, even very briefly, all of these significant results, we have chosen to focus on two open source technologies: [Xskipper](#)<sup>1</sup>, developed by

IBM, and [Kuryr](#), developed with contributions from Red Hat. We detail their technological background and their impact on BigDataStack use cases and, more broadly, on industrial deployments.

These two technologies tackle and solve acute performance problems. Xskipper gives a performance boost for SQL analytics with big data workloads, while the contribution to Kuryr solves an important performance problem with networking. Although they deal with different levels of big data infrastructure, they have a lot in common. First, they both contribute significant performance improvements that directly affect use cases of BigDataStack. Second, both were deployed in production as early as one year ahead of BigDataStack’s completion, and in the course of 2020 they proved their value in real production environments. Finally, both were open sourced and identified by the EU Innovation Radar program as key innovator technologies.

---

<sup>1</sup>The DATA+AI talk that relates to Xskipper does not do so by name. The name Xskipper was coined just after this talk, when the data skipping technology was open sourced.

## XSKIPPER TECHNOLOGY

The Xskipper framework was developed by IBM as an extension to [Apache Spark](#), arguably the most prominent analytics engine for large-scale data processing. Xskipper permits the creation, management, and deployment of data skipping indexes. Its concrete impact is to enable a huge performance acceleration of Apache Spark SQL queries when targeting datasets in an object store.

Let's briefly explain what data skipping is all about. For a given dataset stored in an object store, the initial step is to build the index, or, in other words, to build summary metadata for each object of the dataset. For example, if the dataset has a column that represents the temperature, the summary metadata could include the minimum and maximum temperatures of all the data records contained in the object. This summary metadata is significantly smaller than the data itself and can be indexed and stored in object storage. Once the metadata is indexed, SQL queries that apply a predicate on the temperature column—for example, queries looking for temperatures  $>30^{\circ}\text{C}$ —can benefit from the index by skipping over all the objects whose metadata does not overlap with the predicate (e.g., we can skip over all objects whose maximum temperature is lower than  $30^{\circ}\text{C}$ , since their data cannot possibly satisfy the query).

Beyond this extremely simple example, Xskipper extends data skipping technology in numerous novel ways, such as supporting user-defined functions, handling Boolean conditions over SQL predicates, and adding other indexes out of the box—for example, a Bloom filter. For details about how to reduce I/O and accelerate SQL performance by orders of magnitude



Funded by the European Commission Horizon 2020 Work Programme, the **BigDataStack** project will deliver a complete collection of open and interoperable technology building blocks for a variety of big data stakeholders, including infrastructure operators, application developers, data providers, data scientists, and data consumers. The BigDataStack project combines business knowledge with academic insights, bringing together engineers, developers, and researchers from the worlds of industry and higher education. Overall, fourteen partners are collaborating on the project including Red Hat, IBM, Atos, and the University of Piraeus Research Center.

The BigDataStack is an open technology stack for big data applications and operations. The project is based on a unique data-driven architecture that optimizes big data analytics performance, efficiency, and scalability by dynamically adapting compute, storage, and network resources based on service attributes, data flows, and application interdependencies.

For an overview of the architecture, theory of operation (dimensioning, deployment, operational phases), and use cases see the article [“Researchers and industry deliver open source technology stack for big data applications,”](#) *Red Hat Research Quarterly 1:1*, written by Dimosthenis Kyriazis, Associate Professor, University of Piraeus Technical Coordinator, BigDataStack Consortium.

---

using Xskipper, see the June 2020 IBM blog post [“Data skipping for IBM Cloud SQL Query.”](#)

## KURYR CONTAINER NETWORK INTERFACE

Kuryr acts as an integration bridge between traditional Red Hat® OpenStack® networking and the container community. Red Hat is the main contributor to the project, and during the course of BigDataStack Red Hat developed



performance improvements as well as codebase modernization critical for Kuryr adoption by customers.

Before discussing the contribution of Red Hat to Kuryr, let's address the rationale of running Red Hat® OpenShift® over OpenStack. Deployments of OpenShift clusters over OpenStack are an important part of the hybrid cloud solution. OpenShift is infrastructure independent: it can be run on public cloud, virtualization, bare metal, or anything that can boot Red Hat® Enterprise Linux®. As enterprises increasingly move workloads to the cloud, they still typically keep a sizable on-premise IT stack, which will be part of the infrastructure of a hybrid cloud solution. This positions OpenStack as an excellent solution to deliver the on-premise portion of hybrid cloud, where OpenShift workloads are run over on-premise OpenStack, similar to the public cloud.

Traditional OpenShift installations leverage OpenShiftSDN, which is specific to OpenShift. Using OpenShiftSDN means that your containers run on a network within a network. This setup suffers from double encapsulation, and it introduces an additional layer of complexity that becomes apparent when troubleshooting network issues. Double encapsulation also affects network performance due to the overhead of running a network within a network. In 2018, when BigDataStack was launched, running OpenShift over

an OpenStack installation meant having to suffer from double encapsulation.

Red Hat contributions focused on the Kuryr OpenStack upstream project, which avoids the double encapsulation handicap by allowing direct access to OpenStack networking. Kuryr creates OpenStack resources to provide networking for the Kubernetes/ OpenShift object: it uses Neutron Ports for pods, Neutron networks/ subnets for namespaces, Neutron security groups and security group rules for the network policies, and Octavia load balancers for services. This setup reduces the complexity of the networking layer and also improves network performance. The configuration is detailed in the July 2019 Red Hat blog post [“Accelerate your OpenShift network performance on OpenStack with Kuryr.”](#)

Red Hat made four key contributions during the project:

- integrating Kuryr into OpenShift 4 (i.e., into the Cluster Network Operator),
- modernizing its codebase to better align with the Kubernetes models (custom resource definitions [CRD] based),
- implementing network policy support by leveraging OpenStack security groups (needed to keep feature parity with other SDNs, such as OpenShiftSDN or OVN-Kubernetes), and

- supporting East/West distributed load balancing through OVN Octavia integration, leading to remarkable performance improvements.

The latter was one of the most important contributions. Before OVN Octavia integration, Kuryr was relying on an OpenStack Octavia Amphora driver that required one load balancer VM per Kubernetes service. This led to resource wastage and control plane slowness due to booting up and configuring the VM (unlike just adding a few OVS flows). OVN Octavia integration was key for Kuryr adoption by customers, as it remarkably decreases the resource consumption by removing the need for one specific VM for each service. It also improves the latency/throughput for pod-to-svc communication (up to two times, as it avoids the extra hop on the network to reach the amphora VM), and it improves control plane action performance by one order of magnitude. Because there is no need to create a VM, it achieves a reduction from minutes to seconds.

#### **USE CASES WITHIN AND BEYOND BIGDATASTACK**

The joint use of Kuryr and Xskipper was instrumental in sizable performance improvements. These improvements were demonstrated with a joint IBM/Danaos talk at Think 2019 with the BigDataStack maritime use case, in which BigDataStack algorithms were used to optimize preventative maintenance on Danaos' large fleet of container ships.

The ultimate test came with production use cases. [The Weather Company](#) use case was one of the best examples of the impact of Xskipper. Thanks to Xskipper and complementary technology, SQL queries on a multi-TB weather dataset could be run 40 times faster, while achieving an order-of-magnitude cost reduction at the same time. For more information, see Paula Ta-Shma's presentation at [DATA+AI Summit Europe 2020](#), where this use case and its outcomes are detailed.

The benefits of implementing Kuryr are illustrated in its use by [Services Australia](#). Services Australia is responsible for the delivery of advanced, high-quality, and accessible social, health, and child support services and payments. The organization was interested in utilising the benefits of [running OpenShift on OpenStack with Kuryr](#), primarily by allocating floating IPs to OpenShift services. This feature facilitated inter-cluster communication, in addition to the performance enhancements that come with avoiding double encapsulation. The organization achieved results including better network performance (up to 9 times), streamlined networking, and the potential to expose services externally, without administrative intervention. More information is available on the [BigDataStack blog](#).

## CONCLUSION

We have shown that Xskipper and the Red Hat contributions to the



## Read current articles

## ***Browse back issues***

***Subscribe to print or digital***


***research.redhat.com/quarterly***



# Red Hat Research Quarterly

Kuryr project are two state-of-the-art technologies that significantly enhance the performance of big data workloads. Even though the BigDataStack consortium delivered its complete stack in 2021, this project will certainly continue

influencing big data infrastructure. For example, Xskipper is evolving in various directions, including integration with Red Hat Open Data Hub, and the team is involved in discussions on data skipping indexes for Apache Iceberg. As for Kuryr,

customer adoption is quickly increasing, and new features and enhancements are being requested and implemented, such as support for IPv6 or SCTP protocols and the automatic reconciliation of removed OpenStack resources. 

### About the Authors



**Yosef Moatti, IBM**, graduated from the Ecole Polytechnique of Paris and holds an engineering doctorate from Télécom Paris. He has been involved in cloud technologies for the past ten years.



**Oshrit Feder, IBM**, holds a BSc in Computer science from Tel Aviv university and an MBA from the Technion and has been working with cloud technologies since 2008.



**Guy Khazma, IBM**, is a research staff member in the IBM Cloud and Data Technologies group and has been working on SQL analytics over object storage using Apache Spark.



**Gal Lushi, IBM**, is a research staff member in the IBM Cloud and Data Technologies group and has been working on SQL analytics over object storage using Apache Spark as well as Analytics over encrypted data.



**Paula Ta-Shma, IBM**, is a senior technical staff member in the Cloud and Data Technologies group at IBM Research-Haifa and is responsible for a group of research efforts in the area of hybrid data, with a particular focus on handling big data for both analytics and machine learning.



**Luis Tomás Bolívar, Red Hat**, is a principal software engineer working on the networking team and involved in European research projects.



**Miki Kenneth, Red Hat**, Director of Software Engineering, manages the Red Hat engineering center in Tel Aviv. She is passionate about bringing industry, government, and academic partners together.



**Josh Salomon, Red Hat**, is an architect in the Ceph storage team, with more than thirty years of experience in software development and architecture roles.

```

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\_





## Column

## Building better through research

Collaboration between students and engineers is shaping the future of software, hardware, and as-yet-unimagined complex computer systems.

**About the Author****Heidi Picher**

**Dempsey** is the US Research Director for Red Hat. She seeks out and grows projects with academic and commercial partners in areas such as operating systems, hybrid clouds, performance optimization, networking, AI, and security. As part of the Red Hat Research program, she encourages diverse participation in computer science and engineering research and promotes collaborations with Red Hat researchers.

*by Heidi Picher Dempsey*

I like to read science fiction, partly as an escape from thinking about the complex real-life technology and people we work with daily in the Red Hat Research group. Our research and development projects are fascinating, but some of those problems can curl up comfortably in the back of my mind and wake me at 3 am for a talk. Taking a peek into other technical dreams is a good tonic. In the Afrofuturist *Binti* novellas by Nnedi Okorafor, I was surprised to find instead a reflection of our research group's work in the story of a "harmonizer," or peacemaker, who leaves her home in Africa to travel to an intergalactic university of science and engineering. A math genius and natural builder, Binti is able to generate electrical fields related to math formulas, and she can control currents to alter the world around her. At Red Hat in 2021, we still have to rely on power supplies, hardware, and software stacks, but we are actually changing the world by collaborating with students and faculty to build new complex systems through research.

Creating open source collaboration between engineers and students is key to what we do in

the Greater New England Research Interest Group. Engineers and faculty contribute a deep, first-hand understanding of the many varied and unexpected ways that complex systems fail, and students bring new ways of looking at these systems and new approaches to problem solving. The whole team believes in the beauty of an elegant design but recognizes

---

Creating open source  
collaboration between  
engineers and students is key  
to what we do in the Greater  
New England Research  
Interest Group.

---


that every design carries with it the potential for failure, as complex systems grow and change, and as original assumptions become invalid. We explore these issues in the open and share not only results and solutions, but all the code and details of the environment and

operations that make it possible for others to reproduce the work or alter it for their own systems. This commitment to openness makes the team's contributions more valuable. The National Science Foundation emphasized this recently in a letter to US researchers. The letter highlighted the importance of creating and sharing full datasets and affirmed their interest in supporting new types of data and tools to "lead

the development of next-generation, high-performance networks and computer systems.”

At the human scale, this approach creates organizations like the Red Hat Collaboratory at Boston University and initiatives like Operate First ([see the press release](#)). Students and engineers who are part of Operate First touch all levels of the technology stack, from hardware inventory up through firmware and software compatibility, operational metrics, application performance, and abstract AI modeling of the entire system to help human operators better understand and address complex behaviors.

At the same time, combined university/Red Hat research teams are pursuing projects that reach deeper into the kernel and into special-purpose open hardware programming field-programmable gate arrays (FPGAs) and data processing units (DPUs). (Some of these projects are detailed in [“Research project updates,”](#) Vol. 3:1 of *RHRQ*.) The distinction between software and hardware programming techniques is slowly eroding as these research teams adapt open source tools to create dynamically customized hardware components that could eventually allow us to compile entire systems using only higher-level languages like C.

How do we know this approach builds better systems? Because we’ve discovered and addressed multiple issues with components of complex systems operating in environments like the Mass Open Cloud, and we’ve shared the work in public GitHub issues. The effort has produced software that is more resilient to varied datacenter operating conditions as well as new research publications. How far are we from being able to directly control currents with our fingers? Ask the teams now working on quantum computing, networking, and encryption for that answer. The future may be closer than we think! 



**Project Updates**

## Research project updates

Each quarter, *Red Hat Research Quarterly* highlights new and ongoing research collaborations from around the world. This quarter we highlight collaborative projects in Israel, at Ariel University, Technion University, and The Interdisciplinary Center Herzliya. Contact [academic@redhat.com](mailto:academic@redhat.com) for more information on any project described here.

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

**ACADEMIC INVESTIGATOR:** Prof. Assaf Schuster (Technion)

**RED HAT INVESTIGATOR:** 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 the RedHat® OpenShift® Container Platform and to be applicable to any domain where event-based streaming data is present. Researchers aim to advance the state of the art in the area of complex event processing and to combine academic research with the implementation and deployment of novel CEP mechanisms and techniques.

OpenCEP has reached its fourth and last development iteration. The official release

is currently scheduled for August 1, 2021. In addition to the essential features of a CEP engine, this release will introduce a variety of advanced features, including multi-pattern support, nested operators, execution over uncertain data, adaptive optimization of the pattern-matching process, and a versatile parallel execution model.

**PROJECT: PHYSICS: oPTimized HYbrid Space-time service Continuum in faaS**

**RED HAT INVESTIGATORS:** Luis Tomás Bolívar and Josh Salomon

The goal of PHYSICS is to unlock the potential of the Function-as-a-Service (FaaS) paradigm for cloud service providers and application developers. Specifically, it will enable application developers to design, implement, and deploy advanced FaaS applications in the scope of advanced cloud application design environments, leveraging proven design patterns and existing libraries of cloud/FaaS components.

PHYSICS will offer a novel Global Continuum Layer that will undertake to deploy functions in ways that will optimize multiple application objectives at the same time, such as performance, latency, and cost. PHYSICS will validate the benefits of its Global Continuum Layer and tools in the scope of user-driven application scenarios in three important sectors: healthcare, agriculture, and industry. Visit [physics-faas.eu/](https://physics-faas.eu/) for more information.

The PHYSICS project started in January 2021, with funding from fourteen international partners, and will end in December 2023. It is currently in the phase of defining the architecture and components, as well as the interactions/APIs between them. Researchers are also gathering the use case requirements to better decide on the architecture and the current state-of-the-art offerings and missing functionality that the PHYSICS project will try to address.

## **PROJECT: Kubernetes Optimized Service Discovery Across Clusters**

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

**RED HAT INVESTIGATOR:**  
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 aimed at improving it by making it more balanced, reducing bottlenecks, and improving latency.

The project is now at the design phase in which the different approaches are investigated and discussed, and the API to support them is being formalized. Visit [submariner.io](https://submariner.io) for more information

## **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 INVESTIGATOR:**  
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.

The project coding ended several months ago. As happens often in a large open source project, merging the pull request into the main branch is a challenge in itself. After several iterations of rebase/automated testing, which found a bug in the code due to a rare race condition, researchers


believe they are in the final stages of the merge. While the process is long—each automated test cycle takes days to complete—it was useful.

## **PROJECT: Secured API in Hybrid Cloud**

**ACADEMIC INVESTIGATOR:**  
Dr. Amit Dvir (Ariel)

**RED HAT INVESTIGATOR:** Anna Sandler, mentored by Luiza Nacshon

Researchers will be exploring different types and parts of hybrid cloud APIs and studying the differences between handwritten self-managed APIs and APIs that are managed by API management tools. The goal is to study the base rules of securing a channel-level API in a hybrid cloud and the importance of each component. Using API management tools, we can track data flow in and out of our clouds. By choosing the right rules for a given hybrid cloud—based on data sensitivity, workloads, and user access—data loss, data leak, and other cyber challenges can be prevented.

We are working on improving the API management tool for OpenShift and optimizing its security rules by creating a Rapid PT tool that users can execute to test API security. The tool will demonstrate several possible attacks: DoS, SQL and XSS injections, authentication abuse, and man-in-the-middle (MITM) attacks. Users will get a report from the tests with tips on how to improve their API security. 





# AI ON INTEL®



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

**Learn more at [ai.intel.com](https://ai.intel.com)**