August 2019

RESEARCH QUARTERLY

VOLUME 1:2

Bringing great research ideas into open source communities

How To Train Your Model:
E. Ugur Kaynar's Research Adds Object Store Caching to Ceph, Speeds Machine Learning

Building a Linux Unikernel

Has Open Source Made Patents Obsolete?

Where We Came From: Red Hat Research in Brno, Czechia

Red Hat

RESEARCH QUARTERLY

VOLUME 1:2

Table of Contents

About the Director: 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.

FROM THE DIRECTOR

It is a surprising coincidence, to me at least, that the first two articles in this quarter's *RHRQ* deal with the ownership of information from completely different viewpoints. In the News section, we have an article on technology that is all about keeping private things private.

The ability to use private data without leaking it is absolutely crucial to the pursuit of science and of open source software development, and author Gordon Haff covers a whole series of talks on that subject. Second, we have an interview with Red Hat patent attorney Jared Engstrom, who talks about the role patents played in convincing people to make private things public—in this case, of course, the private things were not personal information, but trade secrets. Engstrom talks about the patent system's role in opening those secrets to public view, which played no small part in the manufacturing revolution in the 19th and 20th century. He also goes into the position Red Hat takes on patents, and how the speed of open source development has made them less relevant in the software industry today.

Our technical articles in this issue, one on object store caching and the other on the development of a Linux-based unikernel, both come out of talks at our first-ever Red Hat Research Day. We brought together 150 professors, grad students, Red Hat engineers, and Red Hat partners and customers the day before our annual Red Hat Summit to hear presentations on the innovative-yet-practical work Red Hat Research supports. The event went so well that we have been challenged to do it again next year, only twice as big—so watch

this space for more information. Our first step will be adding a second event at our Brno developer conference, DevConf (https://devconf.cz), in January 2020.

You might be forgiven for thinking that Red Hat Research began with the first RHRQ issue, shipped just last May. In fact, we have been at this for quite a while, mainly in our Brno, Czechia office—our largest engineering office —where we launched this program almost ten years ago. Brno engineer and manager, Radek Vokal tells the quintessentially Red Hat story of how we got started. Meanwhile, in the first of a regular series from our Research Interest Group leaders, Miki Kenneth talks about the steps they took to get the Tel Aviv office involved with universities there.

Last, but I hope not least, I was moved by the recent, long-awaited closing of the IBM-Red Hat acquisition to think about the platform business—a business we have been in for almost our entire history, but one IBM dominated for decades before that. What kind of research does a platform company do, and who does it benefit? I wrote an essay on the subject for the Red Hat Research team, and they thought it belonged in the magazine—so here it is. I'd love to hear your thoughts on the subject as well (academic@redhat.com).

I have one other exciting thing to mention: You can now subscribe to *RHRQ*! Just browse to research.redhat.com/quarterly to sign up. We will mail you a print copy or email you a link, or both—your choice—and we will charge you only the price of the time required to type your address. Personally, I prefer my magazines in print... showing my age, I know.

I was moved...to think about the platform business—a business we have been in for almost our entire history, but one IBM dominated for decades before that.

—

What kind of research does a platform company do, and who does it benefit?

"Data is useful. It may even be the new oil... But data can also embed information that identifies individuals and their personal characteristics—which they consider deeply private."

HOW DO WE RECONCILE PRIVACY WITH MACHINE LEARNING?

Two critical areas experiencing a wave of innovation and disruption were the focus of the Red Hat Inaugural Research Day in May. The first was data sovereignty in all its aspects (privacy, governance, security) and the second was hardware and microarchitecture innovations for complex, data-intensive, distributed computing systems. There's a clear linkage between these two areas (the former often requires performance provided by the latter), however, they involve distinct communities of experts that don't often interact. One of the objectives of Red Hat Research has been to encourage collaboration across such communities.

This article is focused on the track that covered reconciling privacy with machine learning. These developing technologies are especially pertinent when applying AI techniques to problems where lots of data exists but sharing and pooling that data creates privacy concerns, think medical imaging, for example. In a future installment of RHRQ, we plan to dig into the track that covered the complementary hardware and operating system work needed to accelerating those and other techniques.

PRIVACY AND MACHINE LEARNING: AN INHERENT CONTRADICTION?

Data is useful. It may even be the new oil. Healthcare can use large data sets to correlate different treatment approaches with health outcomes. But data can also embed information that identifies individuals and their personal characteristics—such as some aspect of their health—which they consider deeply private.

Anonymizing the data is one obvious approach. But effective anonymization is surprisingly difficult, given enough data points and the ability to combine additional data from a variety of sources. Furthermore, regulations may place restrictions on sharing healthcare data even if it has been anonymized in some manner.

Alternatively, a company may consider a data set a trade secret and may not want to share it for that reason. That said, anonymization and related techniques are widely used in healthcare and other industries and this is an area of considerable study on its own. (As we'll see when we get to differential privacy)

However, if our primary use of a set of data is to train a model, we have additional options to obscure data. Sharing only the model derived from the training data, rather than the data itself, helps to some degree. But research has demonstrated ways to tease out hidden

data from models derived from it so just sharing the model by itself is not a complete answer.

Three specific approaches were covered in the course of the Research Day morning track chaired by Azer Bestavros of the Hariri Institute for Computing at Boston University (BU): secure multi-party computation (MPC), homomorphic encryption, and differential privacy.

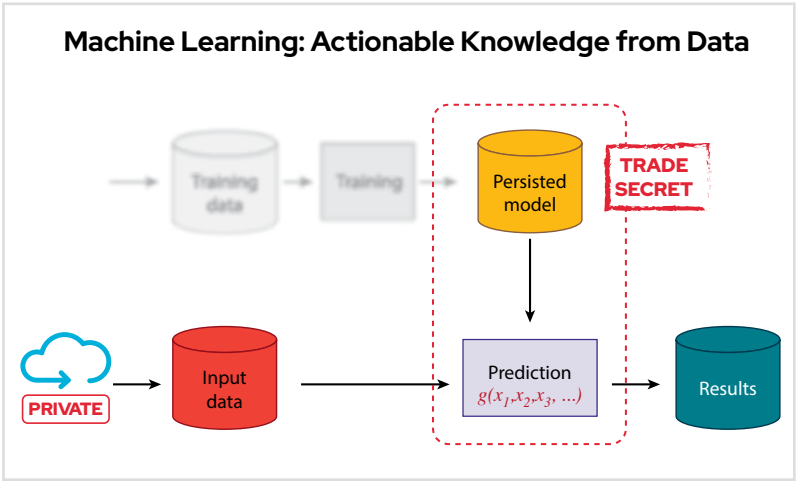
SECURE MULTI-PARTY COMPUTATION

Imagine that a number of companies have data that they are willing to allow the government or other organization to use for some purpose—but only so long as no one else can see their particular data. This was the case with a project done by Boston University with the City of Boston regarding gender-based wage gaps.

Historically, a requirement to keep data secret—even at an aggregated level—would have been a showstopper. Can you share data without actually sharing it? That's where MPC comes in.

The science behind MPC dates back about 30 years but it's become more practical lately because of algorithm and computing improvements. There's also just a lot more interest in figuring out ways to share outputs (such as models) for a data set without either sharing the data itself or involving a trusted third-party.

Essentially, MPC replaces a trusted third-party with a protocol. (In this sense, it has some conceptual similarities with blockchain.) This includes preserving certain security properties, such as privacy and correctness, even if some of the parties collude and maliciously attack the protocol. MPC is an area of active research and there are many variants, especially once one gets beyond "toy" problems. In general, you can think of MPC distributing shares of secrets among the parties doing the computation. No single party can decrypt any of the inputs but all have access to the data outputs (such as with the average salary by gender in the City of Boston example mentioned above).



HOMOMORPHIC ENCRYPTION

While technically distinct from MPC, homomorphic encryption can be used to tackle a similar class of problems. The strongest variation on homomorphic encryption is fully homomorphic encryption (FHE), which allows for arbitrary functions to execute on encrypted data. Homomorphic Encryption is essentially a technique to extend public-key cryptography and was, in fact, first mentioned shortly after the RSA cryptosystem was initially invented.

At Research Day, Kurt Rohloff, CTO and Co-Founder of Duality Technologies, discussed how homomorphic encryption could, for example, allow an owner of sensitive genetics data to encrypt it, send it to some external provider or other organization for analysis, and then receive back an encrypted result—that only they can decrypt. In other words, homomorphic encryption lets a third party perform complicated processing of data without being able to see it.

"Ordinary" encryption schemes make it

"Perhaps we don't need to choose data over privacy if we can apply more sophisticated algorithms to the problem."

virtually impossible for someone without a secret decryption key—such as a cloud provider—to manipulate encrypted data in a useful way. However, homomorphic or malleable encryption schemes allow you to do so. Ensuring such schemes preserve security attributes and allow complex operations is part of ongoing research, as is improving efficiency, since current FTE is too computationally intensive to be generally useful.

DIFFERENTIAL PRIVACY

Suppose you want to widely share statistics over a set of data but don't want to or can't share the individual data points because of privacy concerns?

This need for anonymization, touched on earlier, is hardly new. After all, it's a concern with just about any survey that publishes its results. And it's been "solved" in many different ways—often well enough but vulnerable to known attacks and offering privacy protections that are not well-defined.

In a nutshell, a differential privacy algorithm injects random data into a data set (in a mathematically rigorous way) to protect individual privacy. Because the data is "fuzzed", to the extent that any given response could instead have plausibly been any other valid response, the results may not be quite as accurate as the raw data depending upon the technique used. However, in return, you get results that can't be reverse engineered to derive any specific input.

CAN WE CHANGE THE TRADEOFFS?

Even attributing good intentions to data users—certainly not always warranted—there are tradeoffs between extracting value (from the perspective of both companies and consumers) and maintaining privacy. Perhaps it's not always either/or. But that's mostly the default assumption.

However, we're seeing a growing toolbag of techniques, even if many of them are still squarely in the research phase, that seek to make other tradeoffs instead. Perhaps we don't need to choose data over privacy or privacy over data if we can, instead, apply faster computers or more sophisticated algorithms to the problem.

Visit the Red Hat Research Program.

AUTHORS

– Gordon Haff, Technology evangelist, Red Hat

– Kinan Bab of Boston University covered multi-party computing

Article links:
It may even be the new oil – https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

Red Hat Research Program – https://research.redhat.com/

OPEN SOURCE AND PATENTS: RECOGNIZING INNOVATION WITHOUT BUILDING FENCES

Hugh Brock, Director of Research at Red Hat, had the opportunity to interview Jared Engstrom from the Red Hat legal team. As an IP attorney, Jared manages the Red Hat patent portfolio and has become a specialist in open source patent law.

RHRQ: Can you tell me how you came to open source and intellectual property (IP) law and what you continue to find interesting about it?

Jared Engstrom: Well, honestly, I lucked into it. I started my career at a law firm writing patent applications and then worked at Hewlett-Packard (HP) for several years, a company with a rich history of patents and innovation. In 2012, a recruiter from Red Hat reached out to me with a job offer. They were looking for someone to manage the Red Hat patent portfolio, and it immediately intrigued me. The world of open source wasn't something that I was fully enmeshed in. I had dealt with some open source issues at HP, but open source there was largely a compliance effort, making sure they were managing all their software in compliance with relevant open source licenses. The Red Hat offer was extremely interesting because the relationship between patents and open source is not obvious to most people and, when I tell people I'm the patent guy at an open source company, I get lots of questions: How can that be? How does that work?

I think everyone agrees that the patent system was originally designed to promote innovation and to share ideas. You publish your ideas and you share those ideas with the world, and in exchange for that, we're going to give you something. In the case of the United States, we're going to give you a 20-year monopoly for that idea. There are people today who suggest that the patent system is not doing that as effectively as when it was originally designed, and that the focus for many patent holders is more about excluding the use of new technologies rather than sharing them. One of the things I've discovered is that open source, in some ways, has become the more effective way to promote innovation. A lot of the truly interesting, meaningful ideas, certainly in the software realm, are coming out of open source. And, even though Red Hat may be one of the few "purist" open source companies, all of the big tech companies that are having success, in some way or another, are using open source as a means of doing that.

RHRQ: It had not occurred to me that open source had really picked up that banner or promoted innovation from

JARED ENGSTROM

Manager, IP Attorney at Red Hat

Jared Engstrom is an intellectual property (IP) attorney with a background in electrical and computer engineering. Since joining Red Hat in 2012, Jared has managed Red Hat's patent program, including work on IP issues at the intersection of patents and open source software.

the patent system. It's a very interesting way of looking at it.

JE: Not everyone is going to agree with that, but as somebody who now drinks liberally from the open source fountain, I think the idea has real merit.

RHRQ: In working with universities, how general do you find that understanding to be? I know, from talking to some of my university colleagues, that the old university tech transfer and patent licensing offices are no longer seen as a great way of raising revenue for the university. What do you find the attitude to be there in terms of patenting things versus publishing them allowing them to be free?

JE: I don't have deep connections with folks in the universities to really know where their mindset is on that, but to the extent that I have talked to individuals, I get the impression that there is more interest, priority, and concern over the reputational benefits of doing patent work. If you can get your name on a patent as an inventor, that's good for your own reputation and it's good for the university's reputation. That's because patents are held out as a symbol of innovation, and you can quantify that and measure it. One of the areas where I think the patent system still shines, is that it is the most comprehensive, easily accessible, easily searchable, readily available way to quantify how much innovation is going on in an organization. Whether it's at a university, a company, or wherever, everybody has patents, and you can measure that and have that stand as a proxy for the level of innovation. With the rise of the patent trolls and the disdain that a lot of companies have towards them, the idea of purely monetizing your intellectual property has become disfavored in certain technology environments, such as software. Universities that do a lot of research in the software space are left in a difficult position, because they are doing great research and adding real value to innovation but, because they don't usually productize what they're doing, the options for getting a return on that investment are limited without monetizing the IP—particularly patents—associated with that research. A lot of patent trolls are getting their patents from other operating companies. Historically they have gotten some from universities as well. Some companies are less likely to want to do joint research with universities if they know that those universities are then turning around and trying to monetize those patents via patent trolls. It's a difficult position to put a university in, and, at least at a high level, the approach that Red Hat appears to be taking represents, I think, a different way that may have some long-term advantages for the universities.

RHRQ: You touched on something that resonated with me about the reputational benefits of being on a patent. Certainly being first with an idea is the coin for even getting tenure—it's not merely reputation, it's

how do you succeed at all as an academic? You publish papers or you get your name on patents or "ideally" both. The question is, what can we do in open source to provide that kind of prestige: to credit the invention of a thing, without encumbering it with a bunch of obligations?

JE: I love that question, because I have lots of thoughts on that, even if some of them are not fully baked.I am not a developer per se, although I do have a background in electrical engineering and took coding classes in college and so forth. But I'm on Github and I'm familiar with the mechanics of how repositories work. There are ways of measuring one's level of activity on some of these repositories. I get the sense that there are numbers-based reputational benefits to that. There is sort of the social component of just getting to know other developers and developing relationships and trust and that sort of thing. But I don't get the sense that the reputational benefits have yet risen to the universal level of publications and patents.

On the patent side, how do you provide that reputational benefit while simultaneously avoiding some of the perceived downsides of patents? The way we've tried to approach it at Red Hat is to say, look, there is nothing inherently bad about patents. To me, I often use the analogy of real estate transactions, real property. If you're buying a house, you're buying a piece of property. There is a boundary line around that. That piece of property is yours, you can buy it and own it, and what you do with it really depends on what your priorities are. You can put a big fence around that piece of property, and prevent anyone from going onto it or extract fees from anyone who wants to go onto the property. Or, you can turn it into a public park, and say "I own it, but because I own it, I am going to preserve it as an open space that anyone can use."

In general, the approach that Red Hat has taken with patents is to say "there's nothing wrong with owning patents. In fact, it is good in some ways that we own this intellectual property. But we're going to make them open to the community, and here's all the things that we're going to do to ensure that." We have the Open Invention Network (OIN), which is the world's largest patent cross-licensing organization. Anybody who joins OIN gets a patent license to any patents that relate to Linux. We also have the Red Hat Patent Promise, which is a further assurance that Red Hat is not going to enforce its patents against anyone who makes, uses, or sells open source software. Additionally, we helped form the LOT Network. LOT is an anti-patent-troll network where anyone in the network agrees that if you transfer one of your patents to a patent troll, everybody else in the network gets a license to that patent. So, if you put all these protections around patents, then you can still have the public recognition and reputational benefits of documenting before a government that you have invented something

innovative and useful, and here is the inventor's name on it. If you put the appropriate protections around it, there shouldn't be any downsides to that, either for the inventor or for the company that's doing it. That's the Red Hat approach in a nutshell.

The thing that I go back to again and

> If you put protections around patents, then you can still have the public recognition and reputational benefits of documenting before a government that you have invented something innovative and useful...

again is the analogy of the real estate, the real property concept. By owning that piece of property, you get to control that piece of property's destiny. You can either make it closed or you can make it open. The more patents we can get, the more we can make those spaces open for collaboration within the community.

RHRQ: In your experience doing our agreements with universities to date, do we have a position on patents that arise out of that joint work that we do?

JE: This is still an emerging area in terms of how it might be handled. One approach we have considered is to have both parties agree "in writing" that the priority is to make the research outcomes open and to make any technology developed under the agreement available through an open source license. Specifically, with respect to patents, any patentable ideas developed solely by Red Hat would be owned by Red Hat, any patentable ideas developed by the university would be owned by the university, and any patentable ideas that are developed together, would be jointly owned.

RHRQ: Let me touch on a little different area. We produce a lot of software in these university partnerships. Do we have opinions or a position on the open source licenses that we should use? Is there a preferable one and why?

JE: My understanding is that in true open source fashion, we generally leave it to the individual developers to make the choice about which license makes the most sense. Obviously, there are licenses that offer more benefits in terms of how they apply to patents—certain open source licenses have either implied or explicit patent license grants, the General Public License (GPL) being a primary example of that. However, many developers today prefer the more permissive open source licenses such as Apache or Business Source License (BSD).

RHRQ: Is there any reason that a university should be afraid to work with us in an open source context? What do they have to fear?

JE: Well, I guess maybe I would flip that on its head a little bit, and say what are the advantages of working with us in an open source context? Given the rate and pace of technological change today, especially in software, there are real advantages to working with a company that is adept at adapting to change. I think one of the benefits of getting involved with Red Hat is that we have developed the skills of a surfer. I think Red Hat has become pretty good at identifying those technological waves as they come in, catching them and riding them, and then getting off when the wave is losing energy and preparing for the next one. Innovating becomes less about building static fences around ideas and more about being able to ride waves. No wave lasts forever, but if you can dynamically leverage the energy from one wave into the next, then you create increasing value over time. So I think learning to ride and cycle through those waves efficiently and effectively is a huge benefit to a university.

Article links:
Open Invention Network – https://www.openinventionnetwork.com/

Red Hat Patent Promise – https://www.redhat.com/en/about/patent-promise

LOT Network – https://lotnet.com

OPENING DOORS WITH OPEN SOURCE: HOW WE LAUNCHED THE UNIVERSITY PROGRAM AT RED HAT

Open Source was not always the default choice for software development. Research groups have not always seen the value in opening up not only the code, but also their research data, processes, plans and communications.

They say that change is hard and that universities have a long history and set of experiences that make introducing radical changes "in the way they think and work" even harder. So why did it change, and how have universities started benefiting from the open model as seen from the work Red Hat has done with the Czech Republic?

Red Hat started operating in the Czech Republic in mid-2004. One of the main reasons to open an office in this country was the significant participation of Czechs in various open source projects such as the Linux® kernel, Gnome and KDE desktops, and low-level tools and compilers, such as Apache and GNU Compiler Collection (GCC). Back in those days, all of these folks were either students or post-docs at one of the universities who contributed to open source in their spare time or as part of their research projects.

Red Hat hired them, grouped them together, and started an office that has become a very successful software engineering hub. However, with this approach we realized that we were leaving a gap at the universities themselves. Open source was not then the default choice for development, and most of the faculty still didn't believe that the open model for development and collaboration could work. This has been especially true for the research community who are very protective of their algorithms and data which they think of as a "competitive advantage" over other researchers. Therefore, after forming the office and after two years of growth, in 2007 Red Hat went to the universities to convince them otherwise.

It wasn't easy at first—there were a couple roadblocks that we had to overcome. The first roadblock was that open source was not perceived as a valuable and sustainable development model. Additionally, the universities were strictly opposed to publishing the code of the projects their faculty and students worked on, even when it was an undergraduate thesis or simply a regular small project for their classes. The fear of having someone copy the work of someone else was perceived as bigger risk than the potential benefits of being open.

Another interesting roadblock that

Radek Vokal (center) answers questions from Brno developers and students after a talk at Devconf 2009. The event, small at the start, has grown to 1600 participants last year, the max capacity for the Brno University of Technology venue where it is held.

we hit—one which we hadn't even thought about before—was money. All of the universities were set up in a way that required anyone from the outside wanting to do anything with their students had to pay: per student, per year, or via some fee that always came up in discussions. Our budget was not ready for that.

START WITH THE STUDENTS

So, we started with evangelism. We talked to students we already knew and professors whom we knew from our student days, especially those who had already heard about open source and were open to helping us out. We started bringing interesting people into discussions with students, usually through the professors we worked with. They would let us "hijack" an hour or two of their class and talk about open source and why we think it's the future.

We approached some students with open source-related projects, typically in the form of a master's or bachelor's thesis, that required the student to work closely with an upstream community and contribute back to the project. Of course, because the projects were all open source, the resulting work was something that had to be shared based on the license the upstream project had. I remember that the first project we did this way caused a huge discussion amongst the professors judging the work. It seemed 4 out of 5 of them had no idea how open source worked and they questioned the participation of the student. They asked, "How can they prove this is their work and their ideas?" "Why is he showing us code that someone else had to fix after him?" "This seems like it is someone else's work, not his or hers?" We still had a long way to go, but the students helped us a lot.

Thanks to several professors, especially Vaclav Matyas from Masaryk University and Tomas Vojnar from Brno University of Technology, we kicked off some joint efforts. The whole idea was to hold meetings and bring new ideas and experiences to the students. What we started doing back then we would probably call meetups today, but this was not common 10+ years ago. We organized courses and workshops where students would come voluntarily without getting any academic credit. We started with what we knew best: Linux administration, including going into the details; hacking in C or Python; and even doing simple things like holding

install-fests. Remember, at this time, even installing a Linux distro involved a lot of manual tweaking. We talked about open source principles and licences. The students loved it! We were getting more students in these free course than some professors were getting in their traditional courses.

Based on these courses, we started getting lots of interest from students in how they could work more with us. They wanted to do projects or internships at Red Hat. It was an amazing time. At one point we had so much content that we wanted to share at the universities that we started organizing our own conference, DevConf.cz. The conference started at Masaryk University in 2007 and had approximately 20 sessions of talks and workshops. Back then this was a huge number! In 2018, we had over 4,000 attendees, almost 280 speakers, and participants from all over Europe. The event has since spawned DevConf IN (Bengaluru, August 2-3, 2019) and DevConf US (Boston, August 15-17, 2019).

We built strong connections. Most of the folks we hired through these university activities wanted to contribute back. What started as a small effort by a few people grew exponentially in several directions. All of a sudden, we had courses and labs focused on operating systems-level topics and, thanks to people from JBoss, a new focus on the application layer. Keep in mind, we still had zero funding and this all grew very organically.

At this point, the university representatives realized that there could be benefits for both sides. After a couple years of us approaching students directly, the universities realized that this is what the students wanted. The students desired a close cooperation with experienced people from an international company. Students valued the connections they made through us. We introduced them to big names in the field and we invited visible community contributors to speak with them. This made a huge impact on the universities. All of a sudden, we had several open source projects and research groups approaching us and asking us to help them with open source and how they could collaborate, not only with us, but how they could build a community around their projects.

Stunned surprise: Some talks at Devconf really catch people off guard.

TURNING PROJECTS INTO PRODUCTS

Our first thought had always

Ten years later, the AV is somewhat less of a challenge, but Devconf still requires hours of volunteer time from Red Hat IT.

been to focus on the students and researchers. As long as there was an idea to build something in an open source way and someone had spare cycles to work with that project, we would help. We never thought about these research projects as something Red Hat would immediately benefit from. As it turns out, there was such a wide range of projects that many of them have proven to provide very tangible benefits.

These include projects such as ABRT, Automatic Bug Reporting Tool. Developers who use Fedora might know this tool well. It collects all crashes on your system and reports them back to the project maintainers. The idea for this project had been around for some time, but we never found the time and resources to kick it off. We had a simple proof-of-concept that quickly uncovered that we needed something more scalable. If a kernel crash appears on ten thousand boxes, you don't want each one of them to open a bug report including all the same logs and data.

A research project started at Brno University of Technology works on how to collect and analyze these thousands of reports and find similarities in the core dumps. That required a way to get the most valuable information into a bug report so that a package maintainer could fix the issue. The project is called FAF, Fedora Analysis Framework, and it has been used successfully in the Fedora infrastructure for the last couple of years. It evolved into a server which collects tens of thousands of ABRT crash reports per day and provides fast detection of duplicate problems, statistics, and clustering functionality.

Other projects followed a similar path. We had people using different tools for static code analysis. Most of these tools were proprietary and they had never really been run against large code bases such as the Linux kernel. A few engineers from Red Hat started working closely with a research group called Verifit and improved the performance and the results of their tools based on analyzing the very large code bases that are present in the open source world.

Some research topics were also found by just talking to developers and analyzing the patterns of how they use things. There was one done by a student from Masaryk University which looked at the potential security issues that can happen when a developer misunderstands the

documentation of an API. As a result of this project, the student submitted patches to the OpenSSL project and documented the most common issues that he found.

Last, but not least, a project in a completely different category combined the skills of several different groups. The project is called Robomise.cz and at first glance it looks similar to the various Scratch implementations designed to teach coding. The project brought together work on an infrastructure to run on, a user-experience designer to design a nice interface for it. On top of that there were a few people looking at adaptive learning and how to change the task level based on progress on previous tasks. This involves monitoring the activities performed during the tasks using machine learning to understand how many times a person is making errors or succeeds. This information is then integrated into the UI to advance the user by several levels or roll them back to a previous one if they need to relearn foundational skills. As you see, this is not a project that Red Hat would ever plan to productize and support, but, thanks to this work, we have students who successfully develop on Red Hat OpenShift and others who use it as an AI/ML platform.

There are many more examples of how academic inquiry and the open source approach mutually reinforce each other. And there are new projects starting all the time. You can see the list of active projects at https://research.redhat.com. Feel free to watch their progress, or even contribute, since the work is being conducted upstream in the open source projects themselves. We are always interested in receiving new research proposals that can be supported by our engineers in the Czech Republic, and we're delighted that we've been able to take the approach we developed in Brno and extend it to other universities around the globe.

AUTHOR

– Radek Vokal, Senior engineering manager, Red Hat

Article links:
Vaclav Matyas – https://www.muni.cz/lide/344-vaclav-matyas

Tomas Vojnar - https://www.fit.vutbr.cz/~vojnar/index.php.en

DevConf IN - https://devconf.info/in

DevConf US - https://devconf.info/us

FAF - http://retrace.fedoraproject.org/faf

Verifit - http://www.fit.vutbr.cz/research/groups/verifit/.cs

OpenSSL - https://www.openssl.org/

documented - https://research.redhat.com/academic_pubs/why-johnny-the-developer-cant-work-with-public-key-certificates/

Robomise.cz - http://robomise.cz

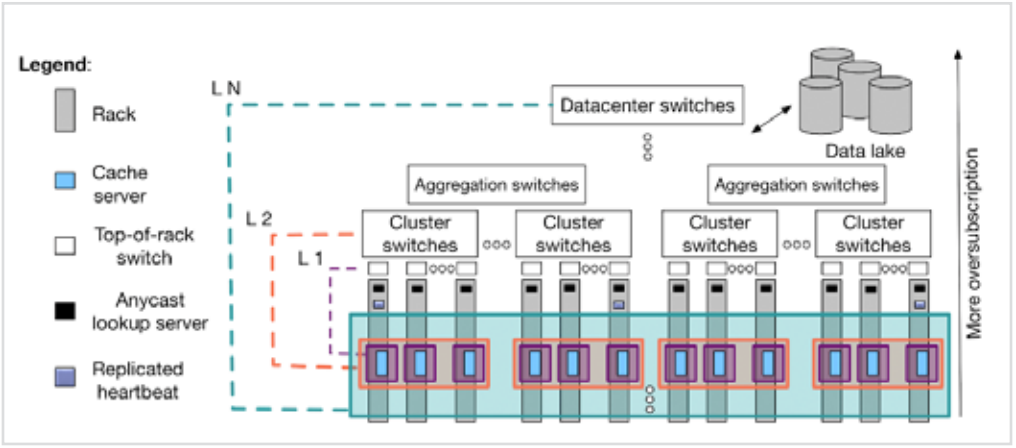
https://research.redhat.com

How have universities started benefiting from the open model, as seen from the work Red Hat has done with the Czech Republic?

COOPERATIVE OBJECT STORE CACHING FOR BIG DATA ANALYTICS

The Datacenter-Data-Delivery Network (D3N) is a novel multi-layer cooperative caching architecture for object stores. The D3N is designed to accelerate big data analytic workloads with strong locality traits and limited network connectivity between compute clusters and data storage. It started as an academic research project, but now the code is "upstreamed" as a Ceph open source project (RADOS Gateway) and will be deployed in the Mass Open Cloud production cluster.

The ability of an organization to make good use of data analytics depends on the amount and variety of data it can collect, and the speed at which it can glean critical insights from that data. Interestingly, these two factors are somewhat in conflict—the more data you collect, the longer it takes to do the data-intensive model training work that is required to gain the insights you need. The caching work we describe below is focused on reducing the problems this conflict causes.



The more data you collect, the longer it takes to do the data-intensive model training work that is required to gain the insights you need.

To help customers do large-scale analytics today, many datacenters include low-cost, centralized storage repositories, called data lakes, to store and share terabyte and petabyte-scale datasets. By necessity in large organizations, distributed big-data analytic clusters such as Hadoop and Spark must depend on accessing a centrally located data lake that is relatively far away. Even with a well-designed datacenter network, cluster-to-data lake bandwidth is typically much less than the bandwidth to storage within the compute clusters. Consequently, users will manually copy a repeatedly accessed dataset to local storage (e.g. HDFS), incurring complexity and performance overhead to manage data placement and replication and to maintain consistency between replicas.

To address these network limitations we designed a multi-layer throughput-oriented caching architecture, Datacenter-Data-Delivery Network (D3N), that uses high-speed storage such as NVMe flash or DRAM to cache datasets on the access side of each layer of a hierarchical network topology.

Red Hat

Such caching allows big data jobs to use the capacity available at each network layer in an oversubscribed network. D3N dynamically allocates cache space to different caching layers to maximize utilization of each network layer.

We implemented D3N as modifications to Ceph's RADOS Gateway (RGW), a Ceph component that supports the S3/Swift-compatible object interface supported by most big data frameworks. D3N requires no changes to the interfaces of any Ceph services, involves no additional metadata services (e.g, to locate cached blocks), and implements all policies based purely on local information.

HOW D3N WORKS

D3N improves the performance of big-data jobs running in analysis clusters by speeding up reads and writes to the data lake. The D3N architecture as shown in the figure above has three components:

- *cache servers*, to which client requests are directed and act as proxies for the back-end object store, storing data locally for reuse
- *lookup service*, used by clients to locate their "local" cache
- *heartbeat service* to track the set of active caches

Cache servers are located in the datacenter on the access side of potential network bottlenecks, and organized into pools of different sizes, with cached data, in all but Level 1, distributed across these pools via consistent hashing. The resulting logical caches form a traditional caching hierarchy, where caches nearer the client have the lowest access latency and overhead, while caches in higher levels in the hierarchy are slower (requiring multiple hops to access), potentially larger (incorporating storage from more individual cache servers), and shared by more clients. The L1 cache server nearest to the client handles object requests by breaking them into blocks, returning any blocks which are cached locally, and forwarding missed requests to the block home location (as determined by consistent hashing) in the next layer. Cache misses are forwarded to successive logical caching layers until a miss at the top layer is resolved by a request to the data lake.

D3N supports:

- *Read cache* where local requests are stored in L1 and global requests are stored in L2.
- *Dynamic cache partitioning algorithm* for partitioning the cache space into layers to minimize mean request latency. The algorithm uses the observed throughput and access patterns to optimize the division of cache capacity between rack-local and cluster-wide data.

The project is the result of two years of collaboration with two universities, faculty, students and software engineers.

- *Write-back* and *Write-through* cache for applications requiring different reliability guarantees; these modes can be selected on a per-object basis.
- *Read-Ahead* and *User-directed* prefetching mechanism to improve the performance of analytic jobs even further.

PERFORMANCE

D3N is highly performant and can support per-cache read speeds of 5 GB/s, fully exploiting the SSDs and NICs in our system. We also demonstrated with large scale datacenter traces that D3N achieves significant performance improvements for realistic workloads—up to a 3x reduction in runtime vs. uncached when bandwidth-constrained.

WHO WILL USE D3N

We are currently working on a plan with Intel to deploy D3N cache infrastructure with high-speed Optane storage in the Massachusetts Open Cloud (MOC) cluster, backed by the 20PB Northeast Storage Exchange storage cluster. We expect the D3N cache to improve big data application performance in the MOC, delivering great value to researchers like me who benefit from data analytics (ML) to drive their research.

Another good use case for D3N is the Cloud Dataverse project, which provides a large data repository for the research community. Between May to July 2019, Dataverse provides with 407,000 files and 8.5 million file downloads. The real time statistics about Dataverse can be found here.

D3N can speed up the download time for these datasets. In addition, the D3N cache will provide a key benefit for enterprises like MOC core partner Two Sigma (https://www.twosigma.com/), which uses the public cloud to run its data analytics cluster. With D3N, Two Sigma can store their frequently accessed datasets near where they are being used for compute, and decrease the requests to their private datacenter back end.

WHERE NEXT

Last June, the MOC storage team presented our future storage vision and ongoing efforts around D3N to the Red Hat Ceph Storage team, as well as Ceph contributors from Intel. During the meeting, we brainstormed a broad research agenda that will shape the priorities of the project moving forward, maybe even launching new collaborations.

The next step for the D3N project is to support a proper persistence layer in the Ceph Rados Gateway that could be used for a write-back cache. Additional areas for follow-up work include the following:

- DAG based prefetching integration of write caching
- Directory based caching and caching for the WAN in cloud environments
- Investigating cache placement problem to determine what, where and when to cache objects.
- Implementing a metadata cache

The D3N project is a project of the Red Hat Collaboratory at Boston University. It started as a research project by Boston University and Northeastern University students. With the help of Red Hat engineers who mentored the research and provided input into the design and implementation of D3N, the modification code was transferred to the Ceph open source community. It is now available on GitHub where anyone can access it to examine or try.

To learn more about D3N project and its architectural details, visit

https://massopen.cloud/d3n/

You can find the GitHub repository here: https://github.com/maniaabdi/ceph.git
The project is the result of two years of collaboration with two universities, faculty, students and software engineers.

Credit goes to:

E. Ugur Kaynar (Boston University)

Mania Abdi (Northeastern University)

Mohammad Hossein Hajkazemi (Northeastern University)

Ata Turk (State Street)

Raja Sambasivan (Boston University)

Amin Mosayyebzadeh (Boston University)

Larry Rudolph (Two Sigma)

Peter Desnoyers (Northeastern University)

Orran Krieger (Boston University)

Matt Benjamin (Red Hat)

Brett Niver (Red Hat)

Ali Maredia (Red Hat)

Mark Kogan (Red Hat)

AUTHOR

— E. Ugur Kaynar, Boston University

Article links:
RADOS Gateway – https://ceph.com/ceph-storage/object-storage/

Mass Open Cloud - https://massopen.cloud

Massachusetts Open Cloud - https://massopen.cloud

Northeast Storage Exchange - https://www.mghpcc.org/big-data-monster-storage-the-northeast-storage-exchange/

Cloud Dataverse - https://dataverse.org/presentations/cloud-dataverse-0

The real-time statistics about Dataverse can be found here - https://dataverse.org/metrics

GitHub - https://github.com/maniaabdi/ceph.git

EMINE UGUR KAYNAR *is a Ph.D. student in the Department of Computer Science at Boston University working with Prof. Orran Krieger in the Systems Research Group. Her research interests lie broadly in the fields of storage systems, cloud computing, and data-intensive computing.*

Red Hat

Unikernels have demonstrated some advantages over traditional operating systems by offering an alternative method for delivering operating system services without requiring a separate operating system.

DELIVERING OPERATING SYSTEM SERVICES WITHOUT AN OPERATING SYSTEM

For operating system engineers, a couple of questions consistently loom large: How much processing speed can an OS facilitate? How much of operating-system-like services are actually needed in a given use case?

There is of course much more to operating-system science besides speed and size, but these two dimensions can limit or expand the scope of an application's viability. The size and speed race is currently playing out in the area of the unikernel.

Unikernels have demonstrated some advantages over traditional operating systems by offering an alternative method for delivering operating system services without requiring a separate operating system. The other advantage that makes them attractive is their customizability. In fact, they are so popular in many important domains, that some propose that they represent a significant challenge to Linux's dominance.

On the contrary, we believe that unikernels' advantages represent the next natural evolution for Linux, as it can adopt the best ideas from the unikernel approach and, along with Linux's battle-tested codebase and large open source community, continue to dominate. A group of researchers from Boston University and engineers at Red Hat have created an early Linux unikernel prototype that demonstrates how some simple changes can bring dramatic performance advantages to Linux unikernel apps. These changes also reduce the footprint and resource requirements of the unikernel, to enable running it on a breadth of devices such as those proliferating in edge deployments. One of the promising use cases is for cloud providers who deploy client workloads in dedicated environments, benefiting greatly from the small memory footprints and short boot times of unikernels. Unikernels deployed within a microVM have shown six- to ten-fold improvement in boot times over containers. Similarly, a micropython unikernel had an image size of 1MB and required only 8MB of memory to run.

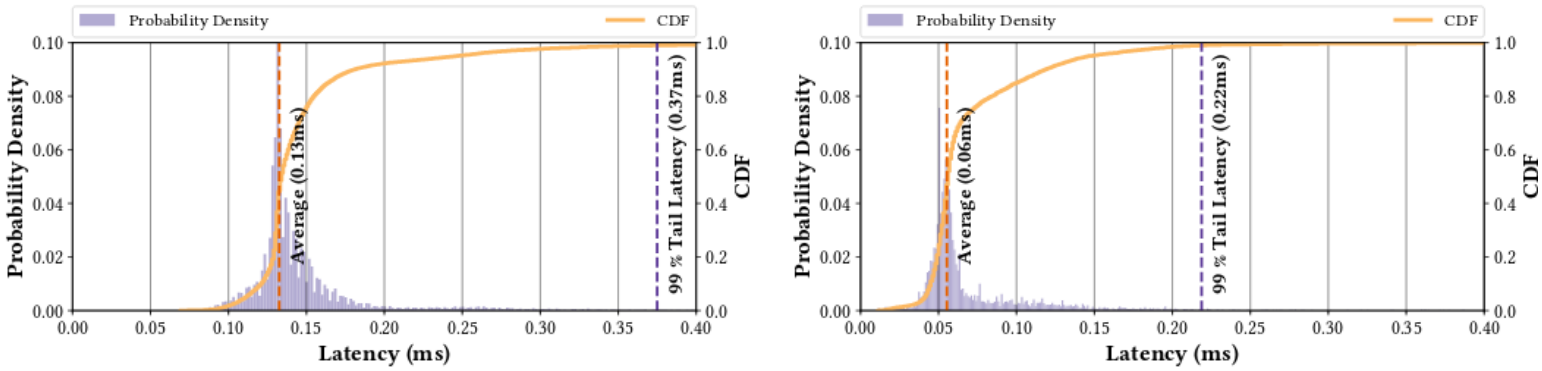
BALANCING SIZE AND SPEED

The approach we took with our research was to turn Linux itself into a unikernel by adding support within the codebase to build a target application into an optimized unikernel binary, while bypassing kernel features that are unnecessary for running the application's workload.

Figure 1 compares the performance of a TCP echo server, written in C, running as a userspace process on Linux with the same TCP echo server linked into a unikernel. In both cases the server is deployed as a VM on QEMU/KVM on a host machine and the client is running on that host machine. While this is obviously a "toy" example, it's encouraging that the unikernel version of the application achieves an average latency half of that of the userspace application and a tail latency that is 41% faster.

where the kernel is statically linked to run a single application. Only with this approach can we enable configure time and link time optimizations that are not possible if arbitrary user- level applications can be run alongside the application we are optimizing for.

Here's an overview of the process we followed for creating the unikernel (UKL) prototype which uses Linux (v5.0.5) and glibc (v2.28).



(a) Userspace Echo server

(b) UKL Echo server

Figure 1.

Protection and isolation of applications provide even more motivation for unikernel research, because an application equipped with a library OS can be made to run in highly-restricted execution domains, such as an SGX enclave or behind a set of software-defined interface.

IMPLEMENTING THE PROTOTYPE

We chose a pure unikernel approach

- We added a new kernel configuration option to specify compiling the Linux kernel as a unikernel.
- We created a call to an undefined symbol (protected by an `#ifdef`) that can be used to invoke application code rather than creating the first userspace process.
- We built a small UKL library which has stubs for syscalls to hide the

ALI RAZA *is a PhD student in the Department of Computer Science at Boston University, working with Prof. Orran Krieger. He is interested in Operating Systems research, and is working on the Linux Unikernel project. He has been working with Red Hat since the summer of 2018.*

PARUL SOHAL *is a PhD student in the Department of Computer Science at Boston University, working with Prof. Orran Krieger and Prof. Renato Mancuso.*

Her interests lie in operating systems, hardware and performance optimizations. She has been working with Red Hat since summer 2018.

details of invoking the required kernel functionality now that the `syscall` instruction is no longer used.

- We changed glibc so that instead of making syscalls into the kernel, it makes function calls into the UKL library.
- We changed the kernel linker script to define new segments such as thread local storage (TLS) segments which are present in application ELF binaries.
- We added a small amount of initialization code before invoking the application to replace initialization normally done by user level code, e.g., for network interface initialization.
- We modified the kernel linking stage to include the application code, glibc, and the UKL library to create a single binary.

WHERE WILL THIS PROTOTYPE LEAD US?

If the unikernel model becomes an accepted part of Linux, our goal would be to enable the same kind of rich optimization that has been achieved in existing research unikernels. For example, one could expose an alternative interface to read which does not dictate the location of the buffer but lets the kernel dictate the location (e.g., in a ring buffer or DMA buffer). Another example of an optimization is exposing a flattened version of the network stack which does not implement the information security aspect of the full stack—if there is only one process, there is no need for privacy and a single ring buffer is sufficient, enabling true zero-copy networking. Once these optimizations are accepted as part of the Linux kernel code, unikernels would be available for adoption in many use cases, from cloud to edge and throughout enterprise IT.

We invite you to watch our progress, or even better, help us improve and further optimize the unikernel prototype. Our project is available on github: https://github.com/razaaliraza/ukl

To learn more, you can find a comprehensive whitepaper here: https://razaaliraza.github.io/papers/UKL.pdf

We would like to thank the core industry partners of Mass Open Cloud (MOC) i.e., Red Hat, Intel, Two Sigma, NetApp and Cisco for supporting this work. Partial support for this work was provided by the USAF Cloud Analysis Model Prototype project, National Science Foundation awards CNS-1414119, ACI-1440788 and OAC-1740218.

AUTHORS

– Ali Raza and Parul Sohal,
Boston University

LIVING TO SERVE: WHY RESEARCH ON A CLOUD PLATFORM MATTERS

A few weeks ago I found myself sitting in San Jose writing a note to the Red Hat Research team about the importance of the software we build at Red Hat, looked at through the lens of our upcoming combination with IBM. Without intending to, I found myself engaged in a defense of the value of Platforms, and the importance of providing one. The team liked it so much that we decided to reprint it, in edited form, here—I hope you will find it as interesting as they did.

My colleague Orran Krieger (https://www.bu.edu/eng/profile/orran-krieger/) of Boston University pointed out to me a couple weeks ago that the people who really made money on the original Internet boom in the '90s weren't the software companies who built the applications that people actually used. The real winners of the Internet gold rush were the networking companies, whose work was defined by the success (or failure) of their implementation of the Internet Protocol. These companies—first Honeywell, then more spectacularly HP and Cisco— provided the platform that everyone had to have to be online. They could partner with anyone to sell their gear and they could support any workload and any application. The enormous campus that Cisco built here in San Jose with all the money they made providing the platform for the Internet, is evidence of the value a good platform provides.

Extra points if your thoughts when reading the above turned to IBM, who provided the ultimate (at the time) compute platform with its System 360 series (https://en.wikipedia.org/wiki/IBM_System/360). IBM, in its heyday, was the perfect platform company—their business was enabling businesses to compete, and they built platforms that would support any workload for any customer. That there was value in this for IBM was undeniable, but the real value was for the customers they served, who could use IBM's platform to do until-then-unimaginable things for their own customers and to make their businesses more effective and more efficient.

I have some personal knowledge of this: My father-in-law Ron Rowe built a company called J.W. Pepper (https://www.jwpepper.com) that relied on IBM minicomputers from the System 3 on to the AS400. In those

Red Hat

Red Hat [succeeded] by providing something with a key difference... A platform based entirely on open source, where customers and partners could not only see how the system was working—they could contribute to making it better.

days, the IBM platform extended to their sales and services organizations—my father-in-law's IBM sales rep had a desk in their office, and he spent his time understanding their business and working out new ways IBM could help them succeed. Of course, this made IBM a lot of money, but it was also an essential ingredient in the success of J.W. Pepper.

It is worth noting that IBM's platform was just that—a thing that you built something on top of. Prior IT vendors who wanted to work with J.W. Pepper came in intending to build a monolithic, bespoke solution for them, and utterly failed. IBM's genius was in refusing to do that. The way Ron tells it, when he asked them to do it, they turned him down flat. Instead, they provided the platform and the training for J.W. Pepper to build its own IT competency to create custom software that was right for its own business. IBM maintained the platform and Pepper maintained the business-specific software that ran on it. With a programming competency far ahead of its competitors, J.W. Pepper went on to dominate the market for choral and band sheet music for the next twenty years.

Today, Red Hat is a successful software platform company, having outlasted a host of other platform suppliers in the enterprise OS market. We arrived at this point by providing something with a key difference from all that had come before: A platform based entirely on open source, where customers and partners could not only see how the system was working—they could contribute to making it better. The success of this model, and of the broader open source movement, has benefited not only end users but also vendors who provide the hardware our platform runs on and the software that runs on top of it. Only the open source development model can keep up with the rapid advances we've seen in the almost 20 years since Red Hat Enterprise Linux first shipped.

So, where does this leave Red Hat Research? We aren't engaged in computer vision, self-driving cars, or protein modeling. Instead, we are focused on areas we think will make our platform—not just RHEL but also the OpenShift container platform and the services supporting it—a better, faster, more capable platform for those who are doing that work. Our mission is to do the research that enables your mission, whatever that is.

AUTHOR

– Hugh Brock, Research Director for Red Hat

HOW A RIG IS BORN - BUILDING A RESEARCH INTEREST GROUP IN TEL AVIV

A Red Hat Research Interest Group (RIG) is a self-organizing team that drives support for and participation in local technical research projects at academic and government agencies or industrial groups.

Every kid in Israel is raised hearing a very famous Israeli children's song written by Yehonathan Gefen named "How A Song Is Born?" that explains to children how a song is being created.

Following the music, I would like to share some key points along with some examples that will give you a glimpse into the journey we have just started in the Ra'anana Red Hat office forming the local Red Hat Research Interest Group (RIG). You'll see how the open culture approach to starting grassroots initiatives works to hone the interests of a diverse group.

FIND YOUR CHAMPIONS

Igniting an initiative is never easy, and even in Red Hat where the open culture is receptive to new initiatives, in order to get the momentum rolling you need to form a coalition. So start small, talk to some people, sell your idea, listen to them, then get their feedback, brainstorm with them, and together form a group that can lead the way. In Ra'anana, the champions of the RIG are a combination of people that really care about patents and innovation and others that are more academically driven. The different perspectives create a nice blend of different ideas and perspectives. Most importantly, the champions are not always the most professionally advanced or technical people. Anyone that is keen to get things done can "push" the initiative forward.

IDENTIFY THE WHY

Hopefully, the initiator of the first steps knows why they are doing it. However, in order to recruit your champions and the rest of the group, the "why" has to be crystal clear. The key point is that defining the group's purpose and motivation, both as a group and as individuals, will help the group perform better in the long run.

For example Red Hat "as a company" has an interest in creating research groups in different locations in order to get exposure to local innovators and prove its interest in fostering new ideas by supporting research projects at all levels. The local Red Hat office, on the other hand, might have a more concrete goal of socializing with the "right" academic schools in order to create a stronger recruiting pipeline in the future, or of developing a curriculum

How is a song born?

How is a song born?
Like laughter
It starts from within
And then rolls out
How is a song born
Like a baby
In the beginning it hurts
But then it comes out
And everyone is happy
Suddenly it walks on its own
How is a song born?
Like a baby

You put three words together
And warm is on a low heat
You run quickly to bring
Some onion from the neighbor
You add two rhymes
A bit of pepper and salt
You mix in three sheep
And throw in a cube of ice

that will enhance the technical skills of the candidates, or of stepping into the academic world from a business perspective, and so on. The people forming the group may have totally different motivations; self-focused or altruistic. They might be using this initiative as a ladder for developing their own career. They might have a special interest in a specific school or project, or maybe they just want to work on a research project. All these motivations are valid, as long as we are aware of them and know how to unite them to drive the group forward.

IDENTIFY THE KEY PLAYERS

Our professional world is changing fast, and the academic world is as well. Schools around the world are reaching out and seeking industrial partners. The trick is to identify key players who are willing to partner and try and form the right partnership. The right partnership consists of good will, mutual agreement on directions and interests, and "of course" technical depth and know-how from both partners. From our local experience, we found that it is sometimes easier to partner with the "second tier" universities (not the three best ranked ones), as these are more open and receptive to new ideas. On the other hand, do not commit to too much. Partnership, is something that you need to invest in, it takes energy, resources and time. Start small, identify a few partners, and invest in creating successful partnerships rather than spreading your resources across many partners.

JUMP INTO THE POOL

Once you have achieved the first three steps, you have to start the ball rolling. Get the champions, have a few basic first steps identified, create some buzz, and initiate the group. Once you have jumped into the pool, and created the group, it will have its own life, it will generate new ideas for the group to evaluate and decide which ones to pursue.

From our experience in Ra'anana, the group came up with several ideas, discussed each one and decided to pursue two different goals. One was to try to attract more students to DevOps track—so one team is working both on the marketing side of the problem as well as on the technical side of preparing the materials for a course for next semester. The other goal was to define a platform (and a set of processes) that will create an easy interaction between project ideas, academic partners, and technical Red Hat mentors. People teamed up on these projects, according to their own special interests, and now there are two separate teams working in parallel.

STEER THE GROUP

Engineers like to innovate, and they like to work on cool new things. New ideas will percolate as soon as you start rolling. You just have to steer the group, work in small steps, make sure the group is working on a regular cadence, don't over commit… and don't stand in the way.

So, I can report to you that the Israel RIG is alive and kicking. It is still in its infancy, but we are already proud. As any parent would teach their child on how to take small steps to create a song, we are confident that the small steps we have taken will produce satisfying results.

AUTHOR

– Miki Kenneth, Research Program Director, Office of the CTO, Red Hat

Article links:
Yehonathan Gefen - https://en.wikipedia.org/wiki/Yehonatan_Geffen

How A Song Is Born? - https://lyricstranslate.com/en/%D7%90%D7%99%D7%9A-%D7%A9%D7%99%D7%A8-%D7%A0%D7%95%D7%9C%D7%93-eich-shir-nolad-how-song-born.html

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

RESEARCH QUARTERLY

VOLUME 1:2

COMING IN NOVEMBER:

- **Research Day 2019, part 2: Gordon Haff looks at unikernels, FPGAs, self-driving device drivers, and much more**
- **Roll Your Own Processor: Building an open source toolchain for FPGA code**
- **DevConf US round-up, plus fun party photos!**

Bringing great research ideas into open source communities

August 2019 – Volume 2: Issue 1

Red Hat