February 2020

RESEARCH QUARTERLY

VOLUME 1:4

Bringing great research ideas into open source communities

Finding Flipper

Newcastle PhDs Georgia Atkinson and Cameron Trotter use deep learning to identify and count marine mammals

Using new tools to analyze old data and improve our picture of the universe

Finding patterns in data, on the fly

Red Hat's Mark Little looks into the future

Red Hat

RESEARCH QUARTERLY

VOLUME 1:4

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

I'm happy to report that our scope at Red Hat Research is now large enough that I am sometimes surprised to learn about things we are doing. In many organizations, this would be a sign of trouble, but in an open source research group it is not just the norm but a sign that things are in good health.

With this issue, I was surprised to discover a common thread: The detection, analysis, implications, and (in one case) realization of patterns. Human beings love patterns, of course–pattern detection is one of the things we are wired for, and I think we find it particularly frustrating when computers fail to recognize patterns that are all too obvious to us. Our research partner Ilya Kolchinsky, of the Technion at the Israel Institute of Technology, has done some very interesting work on Complex Event Processing (CEP) that shows how pattern recognition might be made possible while working on streams of data from different devices overlaid in different ways. This kind of processing is going to become more and more important as the size of data streams increases and I'll be very interested to see how it makes its way into the open source world.

Working on large and growing streams of data is important, but what about data we already have that we've never used before? I'm very excited to see Project Vega coming into its own–this group of Red Hat folks and astronomy PhDs are working to re-analyze old data to update constants in star behavior that we could only guess at before. Working in a similar vein, but on and under the sea, are Georgia Atkinson and Cameron Trotter. Their PhD project at Newcastle

University brings machine learning and big data analysis to photographic and acoustic data to identify marine mammals. Of course, we also have Viktor Malik's DiffKemp project which uses formal methods aided by pattern recognition to analyze differences in Linux kernel versions. What brings all these projects together? The new availability of computing power for machine learning, as well as open source tools to make analysis happen and help us see patterns that we can't with our own eyes.

While we're talking about patterns, I have to mention Christine Flood's open source textile design language project. As our Engineer-In-Residence at Boston University last spring, Christine worked with students to build an open source alternative to the proprietary tools and languages that feed–you guessed it–patterns to looms and knitting machines. Although Christine is a compiler expert for Red Hat during the day, she is a fiber artist in her free time, and this passion project of hers has helped inspire students to continue working to make this textile design language real.

Finally, I talked to our own Mark Little, who leads the technical direction, research, and development for Red Hat JBoss Middleware–that is, when he's not leading research projects at the University of Newcastle. We talked about where Java is headed next, as well as all the recurring patterns we both see in the development of increasingly distributed systems and the tools to make them work for developers. I'll be very interested to see how his ideas on distributed systems take shape in future generations of Red Hat's product suite.

Working on large and growing streams of data is important, but what about data we already have that we've never used before?

...state-of-the-art research projects, while capable of providing excellent results, may at the same time lack scalability and proper software design...

PROJECT VEGA

Scientific computing remains conservative, and researchers, in general, are not keen to trust new technologies. Researchers also don't approach their projects with the engineering mindset. This means that state-of-the-art research projects, while capable of providing excellent results, may at the same time lack scalability and proper software design, resulting in optimization issues, as well as poor performance and user experience. Existing codes were not written with the cloud-native model in mind and are too complex to be redesigned from scratch.

Since any alteration of existing codes is far from being feasible, it is crucial to figure out how we could run them inside a cloud-native environment. This idea originated from early communication with the Department of Theoretical Physics and Astrophysics at Masaryk University in Brno, as part of a Ph.D. thesis titled "New pseudo-rotational stellar atmosphere model." The thesis project turned out to benefit from running in a modern distributed computing environment, and the resulting platform can be reused for any similarly demanding scientific project.

After an initial examination of the available open source solutions (e.g., Red Hat OpenStack, Open MPI, Apache OpenWhisk), we decided on a native Red Hat OpenShift implementation on a Red Hat Enterprise Linux distribution for Project Vega. None of the other software met our standards for this type of computation: it lacked flexibility and scalability. We also adopted hardware platforms, such as graphics processing units (GPUs) and IBM Power Systems, and Infrastructure-as-a-Service (IaaS) solutions from Amazon Web Services (AWS).

Current knowledge of our local universe is based on stellar parameters derived over the last century using astrometry, photometry, and spectroscopy. These parameters cannot be obtained from the observed data directly. Instead, theoretical models are constructed first, and then after finding the matching model, stellar parameters can be found. Alas, the vast majority of stellar models tend to neglect rotation because it was beyond the capacity of traditional computing. We introduced a new model that incorporates existing codes and executes them in a parallel environment with reusability in mind while adding rotation.

We are building a scalable environment that enables us to run parts of our calculation procedure in parallel. We developed a cloud-native application to orchestrate the pipeline and expose

it under the cluster's functionality. The application is designed as a microservice-based architecture. It consists of multiple components communicating with each other via Kubernetes custom resource objects. We defined a new kind of custom resource object that we call "*calculation*" in the scope of our design.

A dispatcher creates *calculation* instances and distributes them to available workers. Workers are copies of a pod managed under a DaemonSet, and they run on the nodes that have the corresponding label, draining all nodes' resources. Generated results provided by individual *calculation* objects are collected by a resource collector and then stored in a network file system (NFS). The garbage collector then deletes the completed Kubernetes *calculation* objects.

The primary objective of Project Vega is to compute the models for the entire grid of scientifically meaningful combinations of input parameters. The upper constraint of the grid magnitude can be set to nine billion combinations of input parameters. A current mid-grade single 8-core server machine would need more than 200,000 years to reach complete coverage of the grid. The cost of running this task on AWS with the highest level of parallelism (m3.2xlarge instance; $0.6/hr) would be approximately 1.4 billion USD. The input parameter space has to be split and delegated to multiple distributed cloud instances flexibly, based on the resources available.

We have completed a proof of concept using a native OpenShift implementation on top of the widely used and accepted scientific codes ATLAS12 and SYNSPEC for the calculation of the stellar atmosphere model against an initial set of parameters to inspect performance and validity of our solution.

Once we reach the production stage, the calculation will be monitored by the tools that already exist in an

The primary objective of Project Vega is to compute the models for the entire grid of scientifically meaningful combinations of input parameters.

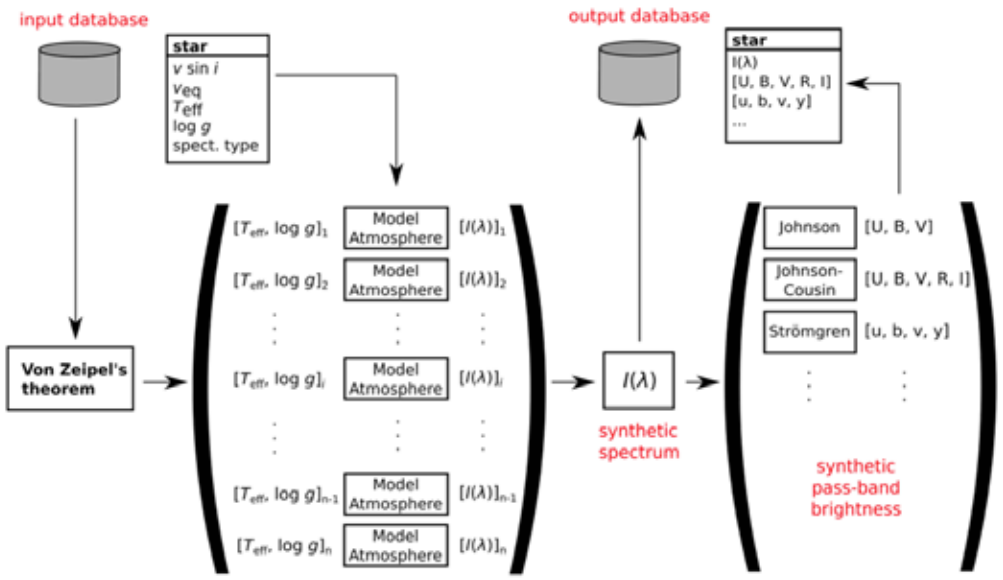


Figure 1: *Rotation added to stellar calculation models*

AUTHORS

Gabriel Szász, Masaryk University

Gabriel is a graduate student at Masaryk University in Brno studying the effects of rotation on measured properties of stars, as well as a as well as a former Red Hatter.

Zdeněk Švécar, Senior project manager, Red Hat

Zdenek has a background in electrical engineering and informatics; he spearheads deployment of regional office infrastructure and technology for Red Hat.

Nikolaos Moraitis, Software engineer, Red Hat

Nikolaos' primary focus at Red Hat is on the developer productivity test platform infrastructure for Red Hat OpenShift. He is an active open source contributor in a variety of projects, predominantly in the Kubernetes community.

Filip Hubík, Software engineer, Red Hat

Filip is a Red Hat OpenStack quality engineer interested in real-world practical applications of cloud computing software, high-performance computing, and data processing and analysis.

OpenShift cluster, such as Prometheus and Grafana, which can send alerts so the team can respond accordingly. Such a framework could be open and publicly shared, built on top of the open source automation we aim to provide.

The benefits of building this framework on open source software are that any institution can contribute to the framework itself or share results and the models can be computed within a realistic timeframe and cost.

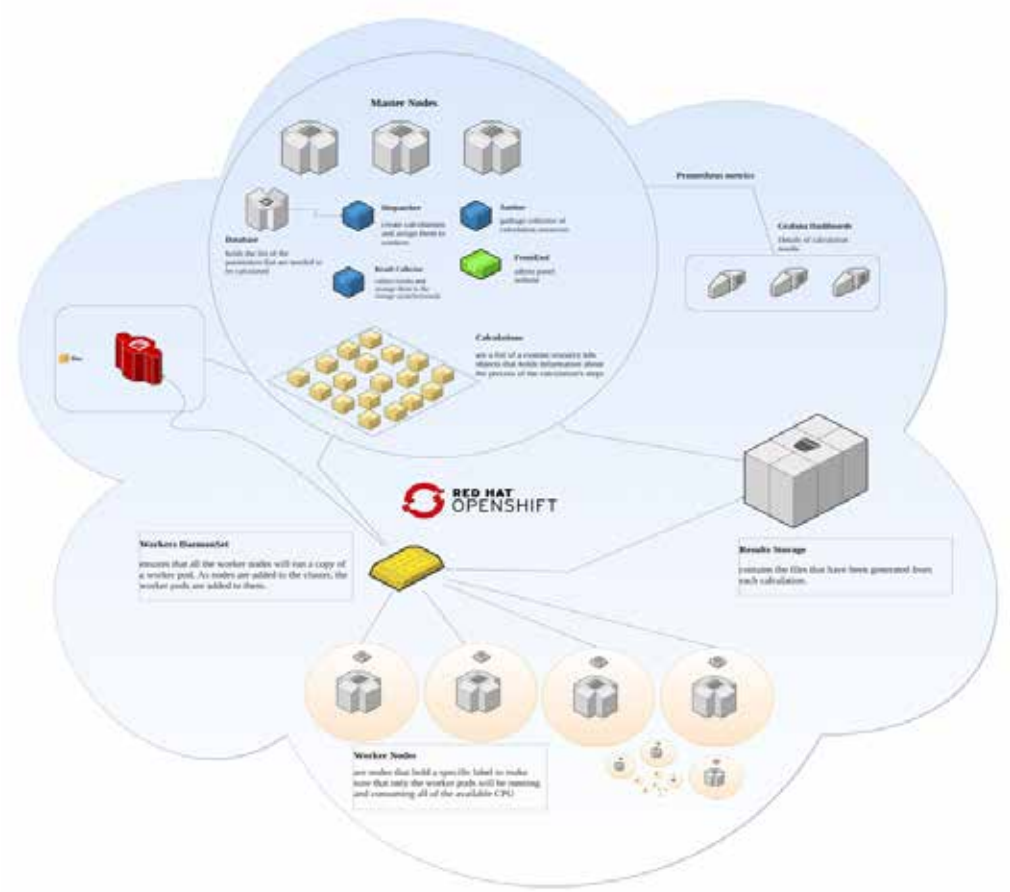


Figure 2: *Project Vega software architecture*

Red Hat

DIFFKEMP: AUTOMATIC ANALYSIS OF SEMANTIC CHANGES IN THE LINUX KERNEL

DiffKemp is a tool for automatic analysis of semantic differences between two versions of a Linux kernel. In order to capture all possible behaviors of the kernel, the comparison is done on the level of the source code: We use a so-called "static analysis." The tool can help to partially automate checking the compatibility of kernel options and the stability of kernel functions, which can make the process of kernel development and deployment more efficient and reliable.

Comparing semantics of two programs is a difficult task—in fact, it is generally undecidable—and is usually solved by complex algorithms that do not scale well for larger programs such as the Linux kernel. However, when comparing two versions of the same software, it is very likely that large parts of the program will remain unchanged and therefore will be not only semantically but also syntactically the same. Moreover, there are a number of code patterns that are syntactically different but have the same semantics that often repeat themselves during software development. Our algorithm takes advantage of these tendencies when it checks for syntactic equality and performs simple semantic equivalence checks by searching for frequent patterns. The main goal of the method is high scalability so that it can be applied to large-scale production software.

WHAT DO WE COMPARE?

To reduce the size of the analysis problem, DiffKemp only checks the most relevant Linux kernel functions, which are conveniently available in the Red Hat Enterprise Linux kernel application binary interface (kABI) list. In addition, it is useful to know if a chosen kernel parameter—either a runtime parameter modifiable via `sysctl` or a kernel module parameter—has the same effect in a new kernel version. Parameters are represented in the kernel code by global variables accompanied by handler functions. To check the semantic equivalence of a kernel parameter, we compare the semantics of all its handler functions and functions that use the associated global variable. Therefore, in general, our method takes two functions as inputs and compares their semantics. The pairs of functions to compare are usually determined by matching their names.

HOW DOES IT WORK?

In order to simplify the static analysis, we do not compare the C code directly. Instead, we use an internal program representation created by the LLVM

compiler, called the LLVM IR. This allows us to leverage the advantages of the Clang/LLVM infrastructure.

The workflow of the algorithm is shown in the following figure:

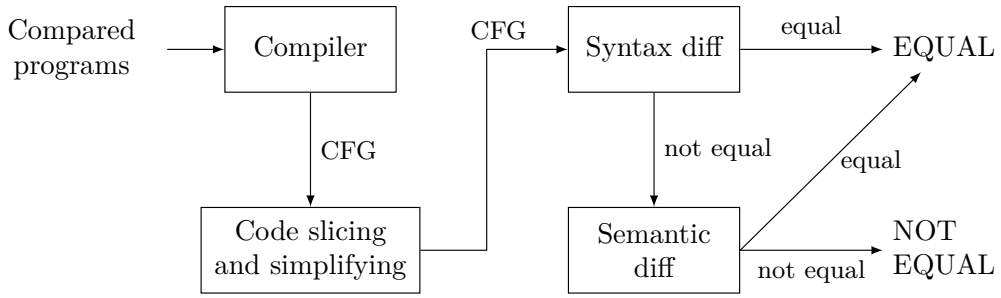


Figure 1: *Algorithm workflow*

First, the compared programs are translated into LLVM IR, which represents each function of the program as a control flow graph (CFG). Next, we perform a series of transformations to remove all information that is irrelevant for comparing program semantics. In the following stage, CFGs are compared for syntactic equality, i.e., whether the CFGs (after simplifications) have exactly the same structure. If this is the case, the algorithm ends. Otherwise, CFGs are compared for semantic equality, i.e., whether the compared programs have the same meaning. The result of the semantic comparison then determines the overall result of the algorithm.

1. Code simplification

The first step of the method is simplifying the compared control flow graphs. This step is performed in order to remove information irrelevant from the point of view of comparing program semantics. By doing this, the following algorithms consume fewer resources and are more likely to prove function equivalence. We run a number of simplifications: For some of them, we make use of code transformations available in LLVM.

- **Dead code elimination.** All unreachable code is removed so that semantic differences that cannot be executed are not reported.
- **Removing kernel debug and warning code.** Contents of printed messages and absolute code location information, such as line number or file name are removed.
- **Removing unused return values.** If the compared code calls a function whose return type has been changed to void and also the actual return value was never used in the old version, the original function is transformed to be void-returning.

1b. Code slicing

When comparing semantics of a kernel parameter, i.e., a global variable, we compare all functions that use the given variable. However, it is sufficient to analyze those parts of the functions that can be affected by the compared variable. In order to ensure this, we use a technique called “program slicing,”

which removes parts of the code that are not related to the property that is being checked. We keep only parts of the functions that depend on the value of the global variable representing the parameter. This is done by computing a dependency relation between program instructions. Both data and control dependency relations can be distinguished. A data dependency between two instructions exists if both instructions access a value and at least one of the accesses is for writing. A control dependency between two instructions exists when the execution of the second instruction depends on the outcome of the first instruction. After computing the dependency relations, all instructions that are not directly or indirectly dependent on the value of the variable may be removed from the CFGs.

2. Syntactic comparison

After performing the simplification and slicing of the compared functions, the next step is a syntactic comparison. We evaluate two functions as syntactically equal if their CFGs have the same structure. This is checked by simultaneously traversing the compared CFGs instruction by instruction, creating mappings between the used variables and memory locations, and checking that the same operations are performed at the same time on the corresponding operands. Here, we leverage the LLVM's Function Comparator module, which is designed for comparing pairs of functions in the same module and used for eliminating equal functions. With some modifications, it is possible to use this component to syntactically compare different versions of a kernel function.

3. Semantic comparison

The last step of the proposed method is a semantic comparison of CFGs. The goal is to determine whether two CFGs represent programs that have the same effect, i.e., whether for the same values of input arguments and equal state of the memory, execution of both functions yields the same return value and ends with equal memory state. There are multiple techniques and many of them use advanced formal method-based approach to prove program equivalence. However, these usually do not scale for large programs, such as the Linux kernel is and therefore we take a different approach. We identified a number of patterns of changes that happen the most often in the kernel that do not change semantics. We implemented an automatic approach to detect these patterns and to determine whether the observed cases actually change semantics. Some examples of these patterns that are the most frequent are (1) changes in layout of structure types, e.g., adding a new structure field may cause other fields of the same structure to get different a memory offset, which does not necessarily change the semantics of the program or (2) moving of code into functions. For the first case, we used debugging information to compare not only compiler-generated offsets of fields but also their names and for the second case, we made use of function inlining available in the compiler.

RESULTS

Although it is too early to show definitive results, we have found—mostly using manual checks—that DiffKemp can correctly identify and precisely locate semantic differences between two Red Hat Enterprise Linux kernel versions in **99%** of cases. Even though most of these changes are intended, e.g., they are security fixes, the tool can be used to locate changes affecting critical parts of the kernel that can potentially break stability and backwards compatibility. Also, the results are promising enough to warrant further study of automated formal verification in this area.

AUTHOR

Viktor Malik, Software Engineer, Red Hat

MARK LITTLE

VP of Engineering, Red Hat

Mark Little leads the technical direction, research, and development for Red Hat JBoss Middleware. Prior to taking over this role in 2008, he served as the service-oriented architecture (SOA) technical development manager and director of standards. Additionally, Mark was co-founder and chief architect at Arjuna Technologies, a spin-off of HP, where he was also a Distinguished Engineer. He has worked in the area of reliable distributed systems since the mid-'80s and has a Ph.D. in fault-tolerant distributed systems, replication, and transactions. Mark is also a professor at Newcastle University and Lyon University.

THE STATE OF DISTRIBUTED COMPUTING 2020

As both a researcher and a technical leader in the Java space, Red Hat VP Mark Little is uniquely positioned to look at the future of distributed computing. We talked with him about what the cloud as a development platform will look like to developers going forward, and where research in this growing area is headed next.

RHRQ: You're a leading researcher in the design of distributed systems and in the way developers use them. With the arrival of things like Kubernetes, serverless computing, and Quarkus, do you think we will finally meet our industry promise of better service with less complexity?

Mark Little: The short answer is no. I've been working in distributed systems for more than 30 years in one way or another. It's a cycle like anything else. Back in the 1980s, we considered large-scale distributed systems to be 100 machines. By the '90s it was up to the thousands. Then the web hit and people were looking at huge distributed systems. And the one thing about the web—and I mean the web, not the internet—that was very different was that it could never have been invented by a computer scientist. It had to be invented by Tim Berners-Lee, who was trying to accomplish a task.

Meanwhile, all the things that *were* wrong with the web are *still* wrong with the web. For example, machines go away and you have hardcoded addresses in URLs and Google is essentially the name service of the world. Computer scientists for years before the web—and even for years after the web—are trying to make things perfect by figuring out how can we do this in an opaque way so that the developer doesn't have to know. We look at things like distributed objects and RPC (remote procedure call) mechanisms to try to simplify things. Theoretically, they're all right, but the problem is we get into this headspace of letting best get in the way of better. Tim Berners-Lee, by contrast, recognized that all this stuff is not going to work properly, but it's good enough.

Five or 10 years ago we had fewer distributed systems. People were thinking about three-tier architectures. Now with microservices, we're talking about breaking them apart and making them into hundreds or thousands of services, and that causes lots of headaches. We'll spend a lot of time—whether it's within Red Hat with our products or upstream in communities—trying to figure out the best way of doing things, and we'll make some incremental improvements. Probably we'll make enough improvements to make it better than it was in the last

cycle. But at some point, the new cycle will come along and we'll all be back to square one in a sense.

What I like to hope at each cycle boundary is, we learn lessons from the previous cycle, and we don't try to reinvent the wheel, which is another problem we have generally.

RHRQ: That sort of brings us to research, which happens to be my next question. What would you say are the promising areas of research today in cloud and distributed systems?

Mark Little: If I was a new student coming into this, there are a couple of areas that would energize me. There's distributed debugging, which is something that we've talked about in the IT community for decades. And it really wasn't that important in the past when people were building monoliths or systems of three, four, or five different connected servers—you could do it by hand. Even today on the web, if something goes down, you can debug it at least to a point where you're satisfied that the problem isn't on your end; it's at the back-end service and then you sort that out.

But now with microservices, where you're basically having to build an application that is many, many different services, you need to figure out where a failure was caused and subsequently how to fix that failure so it doesn't happen again. I think the ability to do this is going to be incredibly important. It has lots of different touchpoints with telemetry and logging. How can we predict what faults may happen in the future based on previous analysis of how faults happened in the past? Sometimes it's not necessarily about being able to prevent a fault. Sometimes it's more about predicting that faults are about to happen and trying to circumvent them; seeing that a service is exhibiting some behaviors that nine times out of 10 in the past meant it was going to fail in the next 24 hours. Therefore, let's not send our requests to that service anymore. Let's route them somewhere else. This is a space that brings in machine learning as well.

Another interesting area for consideration is whether we can learn from life itself how to build distributed systems. We're always talking, particularly in the cloud environment, about how we want our systems to be autonomous and self-managed. Well, the best example of that is our own human cells. We don't sit here thinking, what's my heart doing now to pump blood around my body? It's developed to be autonomous. Can we do that in a distributed system, particularly as the number of components and microservices grows? This is something that people touched on 20 years ago as

If I was a new student coming into this, there are a number of areas that would energize me.

a research topic, but it wasn't really that applicable back then. I do think it might be today.

RHRQ: With the advent of GPUs (graphics processing units) and other kinds of processor architectures, computer systems today are more varied than in the past and will probably become even more so. From an application developer standpoint, what will compilers and tools need to do to take advantage of a heterogeneous hardware base where you're truly not just looking at a CPU (central processing unit) anymore, but rather at a garden of different kinds of accelerators that can be used?

Mark Little: That's a good question. I haven't seen any movement in the research community in that area for a while, but that's because for the last 10 or 15 years, I have been working with Java virtual machine (JVM). There are some good things and bad things about the JVM. The good thing is, you're isolated from the underlying hardware and the heterogeneous aspects that you just mentioned. The bad thing is, you're isolated from the underlying hardware and you perhaps can't get access to some of the nicer performance improvements that are possible with FPGAs (field-programmable gate arrays), or some of the edge aspects of NVRAM (non-volatile random-access memory), or hardware acceleration with different multi-processor capabilities.

I think that if you look at what's happening with Java, that might predict some answers to your question about where the compiler is going. The JVM has been at this level of isolation and it's worked well for a long time. In many ways, it's grown to be an operating system. If you look at what's in the JVM and you look at what you now find in the Linux kernel, there are many similarities. There are threads; there are concepts of processes in some JVM implementations.

But now, look at what's going on with a project called GraalVM coming out of Oracle Labs. They're taking a very different approach: trying to optimize the JVM out of the picture and provide more direct integration with underlying hardware aspects. They provide that by doing compiled Java. So you still write your usual Java—and you can have various plugins—but at the end of the day you can then compile your Java down from the byte code that would normally run on the JVM into a native executable. Doing this gives you incredibly small images because they're essentially x86 code now—not byte code—and that in turn gives you incredibly fast startup times and very high densities. You can start to think about running these applications not necessarily in your traditional desktop CPU, but in an FPGA environment to take advantage of the faster processing available there.

So there's a lot of work going on there, a lot of research. I was at a workshop with Oracle Labs recently, and they're looking at things that are happening in other environments. They're not focusing purely on Java, they're looking at trying to be a generic native runtime for any programming language such as Rust, Erlang, and JavaScript.

If you think about it, certainly the reason that this is interesting from a Red Hat point of view is that it is perfect for Kubernetes. Kubernetes pushes this immutable model where if you change something in your application, you don't change it at runtime. You build another container image to fire off. You don't need the dynamic aspect of Java and other languages in Kubernetes.

RHRQ: You are at the same time a researcher, an upstream community leader, and a product engineering leader at Red Hat. How do you balance the risks that you need to take to do good research with the need to lead and ship a stable product at Red Hat? And does that ever come with a conflict for you?

Mark Little: It does. One of the things I like about open source is that we're able to do long-ish-term research upstream. It's hard for us to talk about research that is three-plus years out upstream because the community tends not to think that way. But it's still possible to do research-y efforts that are six, 12, 18, maybe even 24 months out. I'll give you an example.

We have a product now called Red Hat Automation. Three or four years ago it was two products, Business Process Management and Business Rules Management System (BRMS). Mark Proctor, who is the chief architect for Red Hat Automation, wanted to reset and rethink how to re-architect the two products onto a common code base. And he knew this was going to be a multi-year effort to do upstream. So, we were able to give him that ability and at the same time have his team work on the current product because we've got a vibrant community of contributors outside of Red Hat. This community wants to contribute, but they are not really interested in adding the next bug fix for some large Red Hat customer for instance—that's our job.

However, they are interested if Mark gets out there and talks about "How do we think about combining these two products together? And not only how do we architect a workflow system around a rule system, how do we then present that to users in a graphical manner, whether it's on a browser or within Eclipse?" Those things, even just thinking about them and then talking about them can take months to do, before you can figure out where you go next. But that's exactly the sort of thing that our communities are interested in doing: kicking the tires on some ideas and also maybe even producing some code. So, I think we can do semi-long-term research upstream while still producing our products for our customers and their needs today.

Where we kind of fail as an open source effort—and I don't mean this in a negative way—is I don't think open source works at all in that model for three-plus years out. If you want to do some work on the next funky widget that you know is going to take at least five years to come to fruition, I don't think open source driven by industry alone is going to do it because it's much more immediate than that. However, Red Hat has done this well by working with external academic institutions like Boston University and the Brno (Czechia) universities. We've got a close collaboration with Newcastle University (UK) where I'm based. There are others as well, such as Lyon (France) where we can offer ideas we think are going to be hot in three years' time.

...how do we architect a workflow system around a rule system, how do we then present that to users in a graphical manner, whether it's on a browser or within Eclipse?

...we've been saying for years that it doesn't matter whether it's the cloud or internet of things or microservices or whatever...

So, you're a researcher, you go off and do some research. We might give you some feedback from what we're hearing. Maybe we have to abstract it because it's semi-confidential from this company or that company, but we will give it to you. Then you run with it and maybe some Ph.D.s and some papers come out of it. And then five years down the line when we look at the work, we might think it is now really hot and we want to try to enhance it, bring it into Red Hat, maybe create a new open source project, and then it becomes part of our product portfolio after that. I think we do this well: We combine these different areas of that research and development spectrum quite well.

RHRQ: Tell me about the program at Newcastle. How many Ph.D.s are you working with there right now?

Mark Little: Approximately 45, but we don't fund all those. The model they have at Newcastle at this stage is that they have cohorts. A new cohort starts each year and runs for three or four years—the length of a Ph.D. Each cohort is 12 to 17 people. To support this, there is a collective pool that we fund, along with IBM, Microsoft, and others. We can give ideas for areas that we think are good for this group at Newcastle to do research into, but we don't necessarily say, "Here's our money and it must be given to this candidate to work on this topic." The cohort as a whole helps to collaborate and produce some Ph.D.s that we all benefit from. I kind of like that model.

RHRQ: Java today is not just the most common application development platform, but probably the most enduring one since the mainframe. How do you see Java and the Java ecosystem changing going forward and what are the risks that face it today?

Mark Little: Sun decided in their last 12 months of life to open Java's source code, so they created OpenJDK. Oracle is involved clearly because at the end of the day it's always been a Sun (and later Oracle) technology, so they're still heavily involved in it. But Red Hat is the second largest contributor to OpenJDK consistently year over year. There are other contributors as well: IBM, Microsoft, Amazon, Alibaba. Java is a very vibrant, vendor-agnostic language.

In terms of who's defining the next set of innovative features to go into the language, it's not just Oracle anymore. It's a very eclectic group of vendors and individuals. And I think that's good because it also now helps us in the last couple of years to have moved to a faster release cadence and to start to incorporate new features that other groups have been working on. For example, we've incorporated

Shenandoah Garbage Collection, our garbage collector, which was a much better, more innovative approach to garbage collection than was in OpenJDK by default.

Something else we have been working on since then is the notion of checkpointing the running state of the JVM so you can restart it. Kind of like laptop hibernation, you can checkpoint the JVM, shut it down, and restart it later. And that's going to be good for Functions-as-a-Service and also, hopefully, in Kubernetes environments where we can save a checkpoint of state to an attached persistent volume on a Linux container. So again, lots of great research there. And that, I think, is primarily because we're able to use OpenJDK.

The other thing I would say is when people talk about Java, they often only think about the language, but the language is just one part of it. At Red Hat, we also have a huge middleware stack that builds on top of the language, and the one other point that we hadn't managed to wrestle away from Oracle was the Enterprise Java specifications, which a lot of the world relies on. They were very much dominated by Oracle.

Well, it's taken us two years, but we finally moved all of that from Oracle into the Eclipse Foundation under the "Jakarta" banner. Looking at the Eclipse Foundation now, it's becoming more and more the home for innovative Java efforts. IBM pushed OpenJ9 there now, so that's their JVM. We have Jakarta EE. A lot of the IoT (internet of things) work that's going on upstream globally is based in one way or another on something that's in the Eclipse Foundation. So, it's almost like there's a renaissance of the Eclipse Foundation

we've been saying for years that it doesn't matter whether it's the cloud or IoT or microservices or whatever: Each wave that comes in distributed systems still needs some core capabilities,

and that's really good to see as well.

RHRQ: Related to that, what will application developers gain as Java and Jakarta and the ecosystem grow and progress?

Mark Little: Well, there's one simple thing they'll gain from this: they won't have to worry about Oracle's lawyers coming after them if they use it outside of a certain field. Oracle. in the past. insisted that the application server not be used outside of some particular environments. If you did so, you literally could have an Oracle lawyer knocking on your door. Today, however, Enterprise Java is all covered by open source licenses. You can take various specifications that you may not want to use the whole application server for, pull them out, and use them in whatever way you want. And that's great because we've been saying for years that it doesn't matter whether it's the cloud or IoT or microservices or whatever: Each wave that comes in distributed systems still needs some core capabilities, whether it's reliable messaging or bulletproof security. The Enterprise Java space has defined a lot of these things based on the experience of things that happened before Enterprise Java, so it's not like they built it from scratch, but it's defined these core services and core capabilities that we can leverage in these new waves rather than reinventing the wheel.

I hope that now everything's open and can be collaborated on much more widely, we'll see people not starting from scratch when they need a new reliable messaging service, but building on a spec standard that's already there and actually starting to give more value to developers.

National Science Foundation Awards Grant to Develop Next-Generation Cloud Computing Testbed Powered by Red Hat

Grant to fund creation of national cloud testbed aimed at accelerating innovation in advanced infrastructure technologies

Red Hat, Inc., the world's leading provider of open source solutions, announced that the National Science Foundation (NSF) Division of Computer and Network Systems has awarded a grant to a research team from Boston University, Northeastern University and the University of Massachusetts Amherst (UMass) to help fund the development of a national cloud testbed for research and development of new cloud computing platforms.

The testbed, known as the Open Cloud Testbed, will integrate capabilities previously developed for the CloudLab testbed into the Massachusetts Open Cloud (MOC), a production cloud developed collaboratively by academia, government, and industry through a partnership anchored at Boston University's Hariri Institute for Computing. As a founding industry partner and long-time collaborator on the MOC project, Red Hat will work with Northeastern University and UMass, as well as other government and industry collaborators, to build the national testbed on Red Hat's open hybrid cloud technologies.

Testbeds such as the one being constructed by the research team, are critical for enabling new cloud technologies and making the services they provide more efficient and accessible to a wider range of scientists focusing on research in computer systems and other sciences.

By combining open source technologies and a production cloud enhanced with programmable hardware through field-programmable gate arrays (FPGAs), the project aims to close a gap in computing capabilities currently available to researchers. As a result, the testbed is expected to help accelerate innovation by enabling greater scale and increased collaboration between research teams and open source communities. Red Hat researchers plan to contribute to active research in the testbed, including a wide range of projects on FPGA hardware tools, middleware, operating systems and security.

> "By providing capabilities that currently are only available to researchers within a few large commercial cloud providers, the new testbed will allow diverse communities to exploit these technologies, thus 'democratizing' cloud-computing research and allowing increased collaboration between the research and open-source communities."
>
> MICHAEL ZINKASSOCIATE PROFESSOR, ELECTRICAL AND COMPUTER ENGINEERING (ECE), UNIVERSITY OF MASSACHUSETTS AMHERST

Beyond this, the project also aims to identify, attract, educate and retain the next generation of researchers in this field and accelerate technology transfer from academic research to practical use via collaboration with industry partners such as Red Hat.

Since its launch in 2014, Red Hat has served as a core partner of the MOC, which brings together talent and technologies from various academic, government, non-profit, and industry organizations to collaboratively create an open, production-grade public cloud suitable for cutting-edge research and development. The MOC's open cloud stack is based on Red Hat Enterprise Linux, Red Hat OpenStack Platform and Red Hat OpenShift.

The Mass Open Cloud (MOC) *is a regional public cloud being developed based on the model of an Open Cloud Exchange (OCX), a model where many stakeholders, rather than just a single provider, participate in implementing and operating the cloud. Hosted at Boston University and housed at the Mass Green High Performance Computing Center (MGHPCC), the project is a unique collaborative effort between higher education, government, non-profit entities and industry.*

Beyond creating the national testbed, the grant will also extend Red Hat's collaboration with Boston University researchers to develop self-service capabilities for the MOC's cloud resources. For example, via contributions to the OpenStack bare metal provisioning program (Ironic), the collaboration aims to produce production quality Elastic Secure Infrastructure (ESI) software, a key piece to enabling more flexible and secure resource sharing between different datacenter clusters. And by sharing new developments that enable moving resources between bare metal machines and Red Hat OpenStack or Kubernetes clusters in open source communities such as Ironic or Ansible, Red Hat and the MOC's researchers are helping to advance technology well beyond the Open Cloud Testbed.

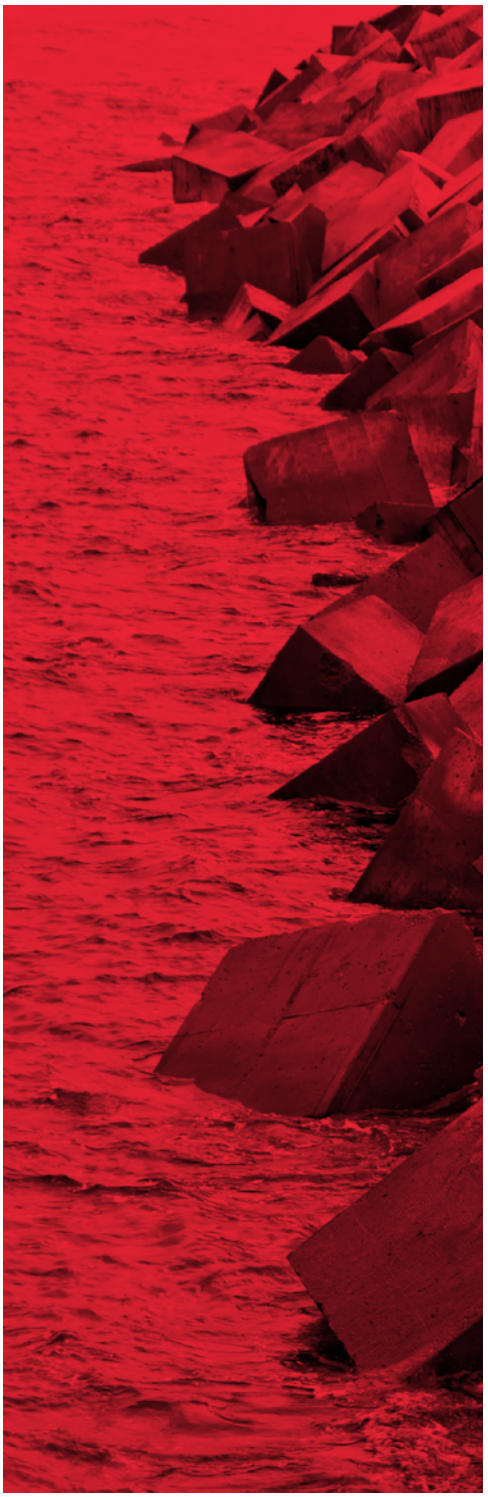
IN SHORT

Boston University, University of Massachusetts and Northeastern University have been awarded a grant from the National Science Foundation for up to $5 million to develop the Open Cloud Testbed, which will integrate the Mass Open Cloud, a project on which Red Hat is a founding partner, with CloudLab.

ACADEMIA AND NON-PROFIT PARTNERS

CORE PARTNERS

CONTRIBUTORS



UNDER THE SEA: DEEP LEARNING IN MARINE BIOLOGY

As global sea temperatures continue to rise, and water pollutants such as plastics and PCBs increase in number along with greater portions of coastal areas becoming urbanized, the monitoring of marine ecosystems is more important now than ever. Cetaceans (dolphins, whales, and porpoises) make prime candidates for monitoring ecosystem change as they are the top predators in the food chain. Their overall health mirrors the current health of the ecosystem. It is a top priority to monitor the cetacean species to assess population numbers, behaviours, and overall health.

Current methodologies for cetacean research, within the field of marine biology, include passive acoustic monitoring (PAM) and photo-identification (photo-id). These methods aggregate large volumes of audio and images which is manually captured and analyzed by experts. This is a costly, time consuming, and error-prone process. The application of deep learning techniques, such as Convolutional and Recurrent Neural Networks (CNNs & RNNs), to both PAM and photo-id methodologies will reduce the number of person-hours needed for species identification and reduce errors in identification due to researcher fatigue.

WHAT DATA IS COLLECTED?

This work is focusing upon two dolphin species which inhabit the North Sea along the Northumberland coastline in the UK mainly bottlenose and white-beaked dolphins. Extensive fieldwork is required during the summer months (weather permitting) for data collection. Fieldwork in the North Sea can be difficult due to working conditions on a small rigid inflatable boat (RIB) that is impacted by weather, sea state, visibility, and lack of shelter from the elements. Dolphins do not have a daily routine so predicting where to search for them is notoriously difficult. Researchers may deploy other techniques to enhance the chance of dolphin encounters.

PAM methods require acoustic recording of vocalisations produced by the cetaceans of interest with devices known as hydrophones. In our work, recordings are collected using SoundTrap ST300 devices. Additional methods that have been used for the collection of acoustic data include:

- Long-term PAM: Between July and October 2019, three hydrophones were deployed in the North Sea which collected over 5 TB of (compressed) acoustic recordings, from a baseline of 16 TB of uncompressed data.
- Acoustic survey using a Wave Glider – an energy harvesting ocean robot. A week-long survey has been carried

out across the Farnes Deep marine region in the North Sea, where white-beaked dolphins are known to inhabit, requiring 24-hour monitoring.

- Fieldwork. Once cetaceans are encountered a hydrophone is placed into the water near the population and then removed at the end of each encounter.

The main method for data collection of above water images is using photo-id methodology with high-quality DSLR cameras. When cetaceans are encountered above water, images are taken from the RIB. Advances in technology have increased the amount of data that is collected during fieldwork. With the advent of smaller and higher-definition underwater cameras, experimentation with underwater GoPro cameras is now used. This allows researchers a greater chance to photo-id individuals from a full body image rather than relying on markings on the fin. The use of new camera technologies, such as those that provide 3D resolution are being trialed. These provide full body scans of cetaceans to more accurately track their health over time.

PHOTO IDENTIFICATION

In use for over 40 years, photo-id methods involve observing from the shoreline or, in a boat. These images are returned to the lab and individual cetaceans are identified based on unique permanent markings such as nicks or scars on their bodies. Cetaceans are usually identified from markings on their fins (as this is above the waterline). In theory any part of the body can be used, but this is the most common, see Figure 1 as an example. If unique features are only prominent on one side of the cetaceans, the task of identification is more difficult. In addition, intra-species cetaceans have very similar markings and body types, making identifying an individual within a pod difficult.

There are systems that help cetacean researchers with analysis, but they require human intervention for final analysis. One example is DARWIN for bottlenose dolphin fins.

Figure 1: *An example photo-id image take from fieldwork. The image has been cropped down to show just the cetacean. Note the prominent mark on the middle left of the fin; this is an easy individual to identify!*

(DARWIN is a software system which allows marine scientists to maintain information for the study of various behavioral and ecological patterns of bottlenose dolphins. The software provides a graphical user interface to access a collection of digital dorsal fin images along with textual information which describes individual animals as well as relevant sighting data. Users may query the system with the name of a specific individual or the entire collection may be sorted and viewed based upon sighting location, sighting date, or damage

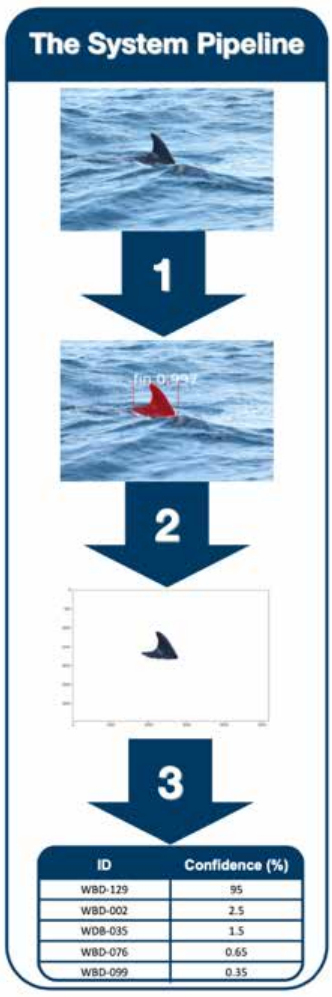


Figure 2: *The proposed outline of four automatic photo-id pipeline 1. Coarse-grain detector; 2. Segmentor; 3. Fine-grain detector.*

category. Alternatively, the researcher may query a database of previously identified dolphin dorsal fin images with an image of an unidentified dolphin's fin. DARWIN responds with a rank ordered list of database fin images that most closely resemble the query image). Source: http://darwin.eckerd.edu/

One of the most advanced opensource software program currently available to cetacean researchers is FlukeBook built upon WildBook. This utilizes computer vision and deep learning to identify individual whales based on their flukes, or lobes of the tail. A limitation of these systems is that they are run in the lab, after data is captured in the field.

Our proposed system allows for detection and photo-id while out in the field collecting data. This system contains three main steps, as shown in Figure 2. First, the images are fed into a coarse grain object detector. This detector is trained to identify dolphins in the image, creating masks of the detections (arrays of Boolean values stating whether each pixel in the image is part of a fin or not). Secondly, these masks are then fed to the segmentor, which separates out any detections into their own images and removes all background. This reduces the computational complexity of processing the images at later stages and removes all noise from the images that is not required for identification. Waves are extremely feature heavy compared to the fins. Third, these segmented images are then sent to a fine-grain detector for photo-id. This stage is still a work in progress, with multiple avenues being explored. Current thinking of how to tackle the problem have led to two possible solutions, one utilising Siamese neural networks and the other utilising autoencoders.

There are currently no known datasets containing images similar to those taken during fieldwork to detect the fins, and no datasets of individual dolphin identifications. In order to train our detectors, a dataset of images was required, so we build our own dataset for analysis. To train the coarse-grain detector, we utilized images from fieldwork performed in Tanzania in 2015. These images were all labelled by hand, drawing masks around the visible fins using the labelling program VIA. This allowed for the coarse grain detector to be trained and fine-tuned while fieldwork was ongoing to build an ID catalogue. With fieldwork completed, the collected images are currently in the process of being labelled. This is to allow for the fine-grain detector to be trained and evaluated.

ACOUSTIC IDENTIFICATION

PAM methods for detecting and identifying cetaceans rely upon high quality underwater recording. The hydrophones we use have a sample rate of 576 kHz, allowing for frequencies of up to 288 kHz to be recorded.

As a point of reference, a typical human can hear frequencies in the 15 to 20 kHz range. This range allows for a variety of dolphin vocalizations to be picked up by the hydrophones, including echolocation clicks and whistles. Echolocation clicks are typically used when hunting prey, and sound like a buzzing noise to the human ear. These vocalizations are not unique to individual cetaceans. Whistles are used for socialising, typically not produced when travelling or hunting. Bottlenose dolphins have been shown to produce what is known as signature whistles, which are unique to individuals. There is also evidence that white-beaked dolphins have signature whistles, however there has not been a lot of research into this area for this particular species.

Current software which aids in the detection of cetacean vocalizations include PAMGUARD and Raven. PAMGUARD is an open-source software which enables users to detect a variety of different vocalisations including whistles, moans and echolocation clicks. Raven Pro enables users to detect different vocalizations, however a fee is charged. Both types of software enable the detection of vocalizations, but not for identification purposes.

When working with signals, or audio, it is typical to transform the inputting signal into an image, known as a spectrogram. This allows you to visually see what you are hearing. In Figure 3, we can see there are two prominent sounds within the audio clip shown by the red lines (red indicates louder sounds, blue indicates quieter sounds). The image shows two dolphin whistles. By collecting these images, we can now use deep learning models such as CNNs and RNNs for image recognition.

This proposed system is made of two stages. The first stage is to detect whistles within the audio. This is a challenging task due to the low signal to noise ratio in most recordings. Low signal to noise ratio means that there is a lot of noise in recordings and not a lot of dolphin whistles. When these whistles occur, due to the background volume of the noise, they typically cannot be seen in spectrograms, as they are too faint. The second stage after detection is to identify the individuals from the detected whistles. This is problematic because at the moment because we do not have a labelled dataset of individual whistles. This requires unsupervised learning techniques to tackle this stage.

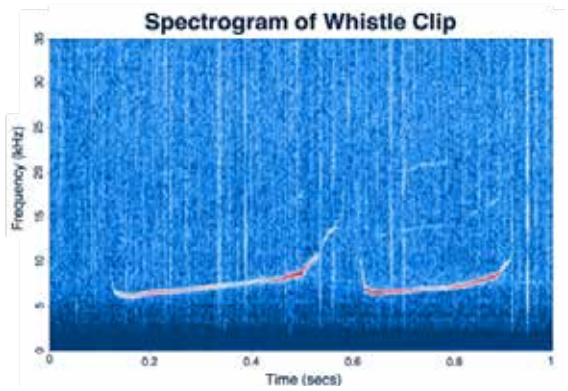


Figure 3: *Spectrogram of an underwater recording*

WHERE NEXT?

Once detection and classification algorithms have been trained and tested, the ultimate goal is to allow researchers to hook their hydrophones and cameras up to a low-cost machine onboard their vessel and

AUTHORS

Georgia W. Atkinson, Newcastle University

Cameron P. Trotter, Newcastle University

Fieldwork

obtain real-time identification of the individual cetaceans they are observing. The plan is to have audio either live-stream from the hydrophone being used (this is not possible with the hydrophones we currently own) or fed into the machine via cables for detection and classification via our models. Images recorded by the attached cameras will be fed directly into our models for on the fly identification. This will require the detection and classification models we are building to be light-weight to achieve real-time detection. Real-time detection and classification will enable marine biologists to get an approximate estimate of the number of cetaceans they are encountering (group size) and who they have in a group. This will enable fast researchers to dedicate a lot more time to research such as health assessments.

We are also building a publicly available dataset known as the Northumberland Dolphin Dataset (NDD). This dataset will provide a new novel computer vision dataset of spectrograms obtained from acoustic recordings and images collected during fieldwork. Unlike other computer vision datasets, the NDD will contain both coarse and fine-grain labels, allowing the dataset to be utilised as a benchmark for different granularities of computer vision research. Images in the NDD are of bottlenose and white-beaked dolphins. These images have been labelled at multiple granularities. At the coarsest level, images are labelled simply as "dolphin". Next, they are labelled as either "bottlenose" or "white-beaked" providing a species level classification. At the highest level of granularity, dolphins that have been able to be identified at an individual level are labelled. This provides a sort of *difficulty scale* to the NDD.

For more information on the work being carried out on the acoustic detection and identification of cetaceans in the North Sea, and how some of the data is collected, please see the short news report presented by the UK's Channel 4.

This work is undertaken within the Cloud Computing for Big Data, EPSRC, Centre for Doctoral Training (CDT) scheme which is supported by RedHat. In the past, groups of students have worked closely with colleagues in Newcastle Upon Tyne to research something of interest at RedHat over a period of two months.

Credit goes to:

Matthew Sharpe (Newcastle University)

Kirsten Richardson (Newcastle University)

Prof Nick Wright (Newcastle University)

Dr A. Stephen McGough (Newcastle University)

Dr Per Berggren (Newcastle University)

MULTI-PATTERN DETECTION OVER STREAMING DATA

Rapid advances in data-driven applications over recent years have intensified the need for efficient mechanisms capable of monitoring and detecting arbitrarily complex patterns in massive data streams, in real time. This task is usually performed by complex event processing (CEP) systems. CEP engines are required to process hundreds or even thousands of user-defined patterns in parallel under tight real-time constraints. Detecting patterns in a stream of events is widely applicable, and the increasing volume of streamed data calls for more effective and efficient approaches to detecting complex, user-defined patterns. Formulating this task as a global optimization problem and applying a combination of sharing and pattern reordering techniques allows constructing an optimal plan satisfying the problem constraints while reducing computation overhead.

Consider a security system in an office building: every room entrance is fitted with a sensor that relays movement detection data to a main controller. That controller is responsible for detecting intruders from the pattern of their movement. Suppose that we are interested in a pattern where an intruder is spotted near doorway A, then immediately passes through entrance B, and finally enters doorway C.

This is a relatively simple pattern, but a significant challenge lies in identifying it as it unfolds in real time from the millions of bits streaming from the sensors. The ability to find patterns in streaming data has a wide range of applications, including financial services, stock exchanges, electronic health record systems, diagnostic images, and the Internet of Things. To a complex event processing system (CEP), all the data items look like events arriving from event sources. Figure 1 shows the architecture of an adaptive CEP system.

Even the most advanced CEP engines can't perform the necessary computations for matching patterns that evolve over time without enormous hardware and energy costs. Our research project was aimed at lowering the barriers to complex pattern matching by formulating this task as a global optimization problem. Our solution was to apply a combination of *pattern re-ordering* and *pattern sharing* techniques to construct an optimal plan. While re-ordering

Even the most advanced CEP engines can't perform the necessary computations for matching patterns that evolve over time without enormous hardware and energy costs.

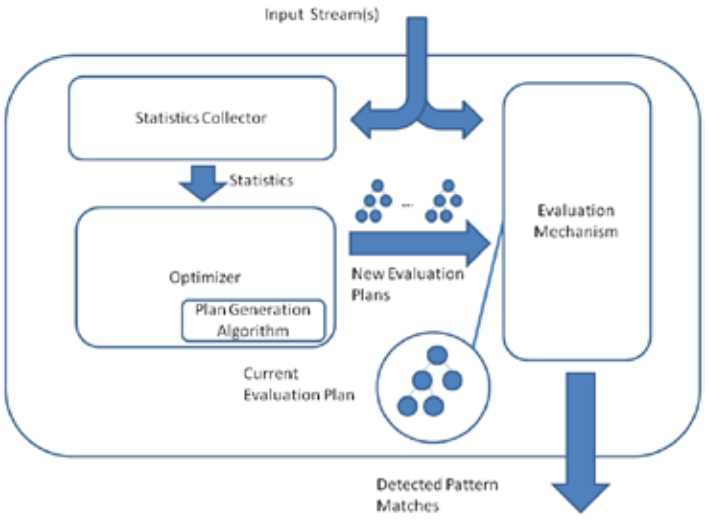


Figure 1: *An adaptive CEP system*

and sharing are common techniques for speeding pattern matching, to the best of our knowledge, no such fusion of these was previously attempted in the field of CEP optimization. We further identified the best possible evaluation plan in the solution space by designing efficient *local search* algorithms that utilize the unique structure of the merged techniques' results.

DEFINING A NEW PATTERN-MATCHING TECHNIQUE

Pattern rewriting methods exploit the statistical properties of the event data to replace the evaluation mechanism with an equivalent yet more efficient one. Pattern reordering is a more specific technique within this category, focused on modifying the order in which the events are processed. For example, let us assume that sensor C generates significantly fewer signals than A and B do. Then, instead of following the order A>B>C specified by the pattern, it would be beneficial to first wait for a signal from C, then examine the local history for previous signals received from sensors B and A. This way, fewer partial matches would be created, resulting in better memory utilization and faster processing of incoming events.

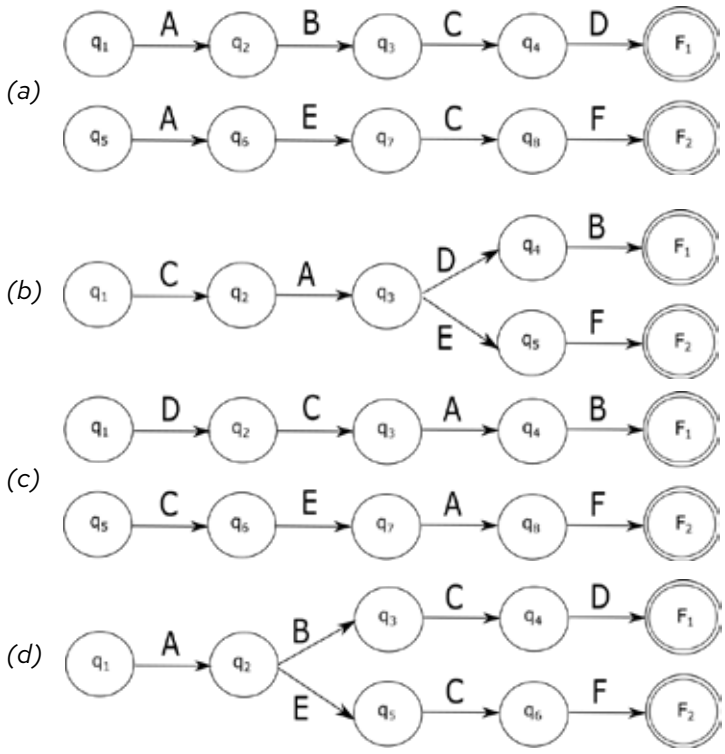


Figure 2: *NFA optimization example for event sequences A,B,C,D and A,E,C,F: (a) no reordering or sharing; (b) reordering without sharing; (c) sharing without reordering; (d) a combination of reordering and sharing.*

The speed and accuracy of pattern detection rely on evaluation mechanisms. The most frequently used one is a non-deterministic finite automaton (NFA). Re-ordering the pattern might give us locally optimized NFAs. However, we created a global shared plan for processing the sensor data by first taking the NFAs and re-ordering them to maximize the common prefix length, and then sharing this newly created sub-pattern. This global plan could never have been created if only one of the two optimizations was employed, or if they were used independently. Figure 2 shows the outcome of the optimization techniques used alone or in tandem.

FACTORING IN TIME

Time is a dimension that cannot be overlooked. The sensor pattern we described above could take a minute or 10 minutes, depending on the individual's pace. Since different patterns may have different time windows, we augmented each state of the multi-pattern NFA with a special time window attribute, set to the largest time window among the patterns

sharing the state. The system uses this attribute to decide whether a partial match has expired. For example, the "intruder" might never have made it to sensor C. Being able to ignore these expired matches further speeds the pattern matching. Our approach was to store NFA instances in a hash table according to their associated state, and the arrival of an event of a given type triggers the traversal of only those instances associated with the sought-for state.

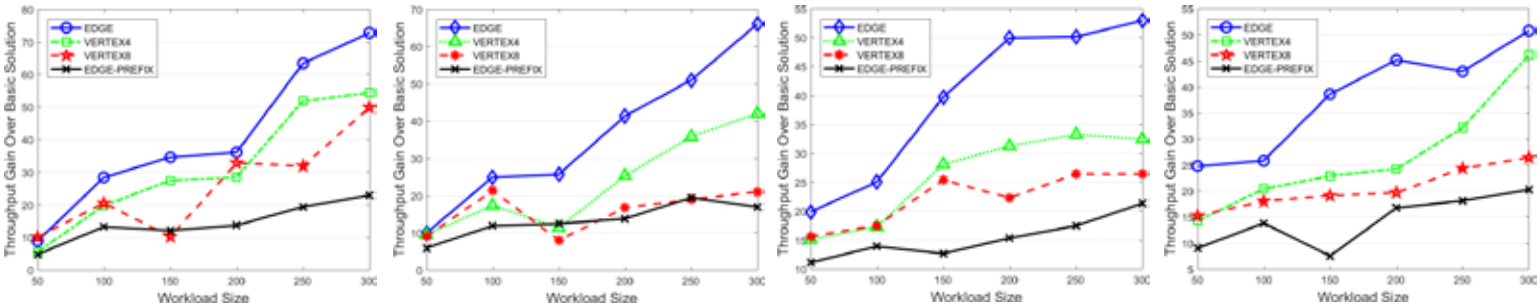
ACHIEVING REAL-TIME PROCESSING

We found that using local search methods had the most benefits for real-time streaming applications. Most importantly, they offer a tradeoff between the quality of the returned solution and the running time of the search. Since the local search procedure keeps a "current best" solution at any point of its execution, it can always be interrupted due to expired time limit and will return a valid solution, albeit not necessarily the cheapest. This property makes local search methods an attractive choice for targeting the CEP optimization problem under tight real-time constraints.

THE PROOF POINTS

To test our solution, we ran several experiments using two independent datasets. The first was taken from the NASDAQ stock market historical records. Each data record represents a single update to the price of a stock, spanning a 1-year period and covering over 2100 stock identifiers with prices periodically updated. Our input stream contained 80,509,033 primitive events, each consisting of a stock identifier, a timestamp, and a current price. We also augmented the event format with the precalculated difference between the current and the previous price of each stock. We considered updates of each stock identifier as events belonging to a separate type.

The second dataset contains vehicle traffic sensor data provided by the city of Aarhus, Denmark, collected over a period of 4 months from 449 observation points. This input stream contained 13,577,132 primitive events, with attributes including the point ID, the average observed speed, and the total number of observed vehicles during the last 5 minutes. The patterns created for this dataset were based on normal driving behavior, where the average speed tends to decrease as the number of vehicles on the road increases. We wanted to detect the deviations of this pattern, that is, violations of this model



(a) (b) (c) (d)

Figure 3: *Throughput gain as a function of the workload size for different combinations of a meta-heuristic, a neighborhood function, a subexpression sharing strategy and a dataset: (a) stocks dataset with simulated annealing; b) stocks dataset with tabu search; (c) traffic dataset with simulated annealing; (d) traffic dataset, tabu search.*

ILYA KOLCHINSKY *is a research scientist with Technion, Israel Institute of Technology, specializing in the various aspects of optimizing event-oriented data stream processing systems. He has a PhD and BS in Computer Science, both from Technion.*

His research interests include stream processing optimization, distributed event-based systems, scalable processing of streams containing uncertain data, approximate complex event processing, stream mining, elastic stream processing, anomaly detection in data streams, and incorporating state-of-the-art machine learning techniques to extend the functionality and improve the performance of currently available stream processing solutions.

where combinations of three or more observations with either an increase or a decrease in both the number of vehicles and the average speed.

Overall, all combinations of pattern re-ordering and pattern sharing demonstrated more significant throughput gains for larger workloads, ranging from a factor of 21 to over 72 when compared to NFAs that used just one technique. Our optimizer achieved the best overall speedup, in some cases up to three times better than that of the runner-up solution. Taking advantage of the reordering opportunities drastically boosted CEP evaluation. Our approach also exploited sharing opportunities when possible, which allowed our optimizer to outperform the pure reordering algorithm for large pattern sizes.

Our experimental evaluation demonstrated a significant performance boost as compared to state-of-the-art CEP techniques. Our next research will tackle additional challenges to CEP such as SLA support and dynamic workload modification. Meanwhile, you can read the linked articles for the detailed explanations of our NFAs, our search algorithms, and our experiment specifications.

I wish to give credit to my advisor at Technion, Israel Institute of Technology, Dr. Assaf Schuster. The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 688380 and was partially supported by the Israel Science Foundation and by HPI-Technion Research School.

AUTHOR

Ilya Kolchinsky, Technion, Israel Institute of Technology

PROFILE OF AN ENGINEER-IN-RESIDENCE: CHRISTINE FLOOD

The Engineer-in-Residence for the Red Hat Collaboratory—a partnership between Red Hat and Boston University—is a great opportunity for engineers to tie their professional experiences to their passions. In Christine Flood's case, the program, which advances research and learning in emerging technologies, brings together her lifelong interests in textile crafts, seeking tech solutions to market-entry barriers, and being at the forefront of Java's evolution in open source.

Today, if you buy a high-end loom or knitting machine, it comes with proprietary software. Similar to how the PC market was 20 years ago, you are basically locked into the proprietary software for the machine that you bought. Christine's passion lies in creating an open source textile design language and eventually fostering a connection between artisanal knitters and weavers and the industrial textile manufacturing sector. The hoped-for result would be more access to markets, creative designs, and the ability to customize for sizes, shapes, and other wearer requirements.

Through the Engineer-in-Residence program, Christine was able to meet with two undergraduate students for two or three hours a week to create their first textile design language prototype. The goal was to expose the students to the Java programming language and other open source software development tools such as GitHub. At the end of the semester, the students got to work together on a project and had grasped some understanding of the open source development model. Additionally, as part of the BU Dolphin Tank, they got an opportunity to share more about their projects with colleagues.

The Engineer-in-Residence program has since inspired Christine to do more work with students because she believes that incorporating fresh ideas from students helps to broaden our understanding of different topics. In addition to bringing different perspectives to a project, Christine remarked that students tend to challenge her assumptions in a constructive way and that while working through a project with them, we can figure out how to make things more efficient.

The Engineer-in-Residence program has since inspired Christine to do more work with students because she believes that incorporating fresh ideas from students helps to broaden our understanding of different topics.

While Christine's career is in programming languages, the Engineer-in-Residence program dovetails nicely with her passion for textile design language. She has a grand vision of an open source textile design language from Red Hat—but in the meantime, programs such as the Engineer-in-Residence allow her to come back to work recharged.

RIG LEADER'S PERSPECTIVE: EXPANDING COLLABORATIVE RESEARCH

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.

Over the last 13 years, what was once a simple reluctant handshake with a single university has grown into a robust, multifaceted and complex program that has the potential to be applied anywhere else in the world. And applied it was; the Czech Research Program has inspired many efforts all over the world, multiple best practices have already been "exported" to other offices in Boston, Tel Aviv, Beijing and Pune where they have grown their own roots and started lives of their own.

SO WHAT MAKES IT SO SUCCESSFUL AND WHY IS IT DIFFERENT FROM OTHER RESEARCH EFFORTS?

Well, firstly we didn't build our university relationships on funding, but on trust and open collaboration. It doesn't mean that money isn't involved; with collaboration this size, it has to come in eventually, the difference is what role it plays. Instead of simply granting funds to the university for a specific type of service, the cooperation is based on common interests that actively involve both sides of the relationship. In other words the content comes first and resources come second. Both are important but the order matters as it determines the nature of the cooperation.

Secondly, we didn't put all our eggs in one basket. Of course, this metaphor deserves further elaboration, as its meaning is twofold. Although in the beginning, it is a desirable modus operandi, we didn't remain concentrated on one university. Every university has its strong fields and areas with enthusiastic people, eager to cooperate and it is important to leverage efforts that make the most sense for both sides of the cooperation. On top of that, healthy competition (whether between companies or between universities) can further reinforce the quality of cooperation in general. Which leads us to the ultimate achievement - connecting multiple universities and/or companies in a joint research project.

As Kamil Dudka, the technical consultant for the Technology Agency of the Czech Republic (TACR) grant to Masaryk University, BUT, Honeywell, and Red Hat notes, "Thanks to the AUFOVER project supported by TACR, we were able to significantly extend the collaboration with Brno universities in the area of formal verification. We need to use advanced technologies to ensure correctness of the software that we distribute while

universities are interested in applying their formal verification tools on enterprise software. The fact that one of the tools (Symbiotic) won the Software Systems category of the International Competition on Software Verification (SV-COMP 2020) suggests that we had selected the right tools to experiment with. Many paths lead to cooperation."

At the same time, we didn't concentrate on one type of cooperation, but developed multiple efforts that each serve their own purpose, however they reinforce each other. While teaching at the university, leading workshops and open source projects can motivate students' interest in internships, an intern's involvement in the workflow of a real product can lead to an interesting practical thesis, a successful thesis may lead to a basis of an interesting dissertation, a supported Ph.D. student may end up involving multiple Master students and leading their individual theses, which can lead to a possible applied research cooperation and grant application. This network of efforts provides students with a plethora of possibilities of involvement and time investment, which doesn't tempt them to leave their studies - in most cases, it ties their practical involvement more closely with their studies, entangling the two into fruitful synergy.

The highest level of cooperation is not based on pulling good students from universities, on the contrary, it is in supporting good students to stay in the academic field and continue doing good applied research - the open way and in cooperation with Red Hat. The majority of students will not continue to their Ph.D. However, Red Hat can be integral in recognizing the best candidates and providing support that will encourage them to use talent the best way possible.

> ...an intern's involvement in the workflow of a real product can lead to an interesting practical thesis...

FROM OPEN SOURCE TO OPEN DATA

And last, but quite the polar opposite of least, Red Hat comes with the open approach. Not only does it allow us to do research in a much more collaborative way, but further accelerates the application of research outcomes by not only opening up know-how, but also valuable datasets, that can be very hard to reproduce. The same approach that has allowed the open source business model to thrive can be applied to the academic world. When applying research models to products or workflows in Red Hat's business, it can provide unaltered and uncensored case studies that allow researchers to make improvements based on real data.

Martin Ukrop, one of the Ph.D. students whose research is supported by Red Hat, is convinced that "applying the open data principles to academic research leads to reproducible results – the measurements, survey datasets and study designs are all publicly available. This increases trustworthiness and eases follow-up works standing on previously published conclusions."

OPEN SOURCE LABS ANCHOR COLLABORATION?

Given the large network of efforts being done at a particular university, the labs are simply just another piece of the puzzle - a platform that allows partners to cooperate more effectively. It provides additional space for teaching, hosting events, and a place for students to meet collaborating engineers and a quiet space to work on research without having to travel to a Red Hat office. We don't need to remove every hurdle from the students' lives, in truth, it's the adversity that makes us strong, but that doesn't mean we can't make things simpler.

AUTHOR

Matej Hrušovský, University Program Manager, Red Hat

Milan Brož, Principal Program Manager, Red Hat

Start using AI in your cloud !

workload prediction &
resource optimization

capacity planning

on-prem & public cloud
cost reduction

Kubernetes
OpenShift

PROPHETSTOR

visit us at
www.prophetstor.com

RESEARCH PROJECTS UPDATE

Faculty, PhD students, and U.S. Red Hat associates in the Northeast U.S. are collaborating actively on the following research projects. This quarter we highlight collaborative projects at Boston University (BU), Northeastern University, Harvard University, and the University of Massachusetts. We will highlight research colloborations from other parts of the world in future editions of the Research Quarterly. Contact academic@redhat.com from more information on any project.

ACADEMIC INVESTIGATORS

Honeywell (Mgr. Tomáš Kratochvíla)

Paradise (doc. RNDr. Jiří Barnat, Ph.D.)

Formela (doc. RNDr. Jan Strejček, Ph.D.)

VeriFIT (prof. Ing. Tomáš Vojnar, Ph.D. and Ing. Aleš Smrčka, Ph.D.)

RED HAT INVESTIGATORS

Kamil Dudka

Ondřej Vašík

PROJECT: AUFOVER (AUTOMATED FORMAL VERIFICATION)

Symbiotic and DIVINE are now available as RPM packages, which makes it easy to experiment with the formal verification tools locally on any system running Fedora. As the next step, we are developing converters of their output so that the results of formal verification tools can be processed by the same utilities and services as the results of static analyzers. Finally, we are developing our own benchmark to further automate re-evaluation of these tools on updates.

The goal of the AUtomation of FOrmal Verification (AUFOVER) project is to develop automated formal verification tools and integrate them for industrial use. The tools to be developed or improved within the projects are Verification Server, Verification Server Client Application, csmock plug-ins, DIVINE, Symbiotic and Testos. The purpose of the grant is to finish the development of university tools based on formal mathematical methods and their transfer to a commercial environment, including integration with industrial partners' tools and incorporation of these tools into the commercial processes for software verification.

ACADEMIC INVESTIGATORS

doc. Ing. Miroslav Bureš, Ph.D.

RED HAT INVESTIGATORS

Štefan Bunčiak

Miroslav Jaroš

PROJECT: QUALITY ASSURANCE SYSTEM FOR INTERNET OF THINGS TECHNOLOGY

The project team has successfully submitted three patent applications in the IoT field. Furthermore, they have released a feature-complete version (2.0) of the open source project, PatrIoT. Currently, they are actively looking for customer use cases on which to base further research and participants in a pilot usage program.

The main goal is to design, implement and verify a framework for quality assurance of products based on the Internet of Things concept. The aim of the framework is to help individual IoT projects to establish an efficient testing and verification strategy of the infrastructure. The proposed framework is based on a model of the IoT infrastructure, composed of methodological part, driven by university team, and a technical part, mainly worked on by Red Hat engineers. The design of the framework aims to be compliant with a continuous integration approach emphasising automation of testing and the quality assurance process.

ACADEMIC INVESTIGATORS

doc. Ing. Ondřej Žižlavský Ph.D

Eddie Fisher MSc., Ph.D

Tetyana Shpilka

RED HAT INVESTIGATORS

Marcel Gazdík

Vojtěch Sokol

Tomáš Meszároš

PROJECT: INNOVATION SCORECARD

The Brno University of Technology (BUT) and Red Hat have concluded a project-definition workshop and launched the initiative. The first sub-project, Atomic Host, an automation container build process, has been completed. The second project, Continuous Integration, started in August 2019. Articles will appear in early 2020 in the Czech CAFINews journal and the Association for Project Management (APM) in the United Kingdom.

Innovation Scorecard is a conceptual performance measurement and management control framework specifically designed for work activities that relate to innovation. Its origin lies in the outcomes of a primary research project that was supported by the Czech Scientific Foundation during 2013-2015. This project's objective was to ascertain whether organizations in the Czech Republic measure the efficiency of their innovations and what metrics they applied to measure these. Results confirmed that organizations that constantly managed innovation were engaged in identifying performance measurements to determine the level of value and benefits associated with innovation. When applied appropriately and in accordance with existing company strategy, marketing drives and HR/corporate policies, processes and procedures, innovation metrics provide managers and employees with opportunities to plan, organize, monitor and control all innovation activities for the benefit of the organization they work for.

ACADEMIC INVESTIGATORS

Bestoun S. Ahmed Ph.D.

Jan Richter

RED HAT INVESTIGATORS

Cleber Rosa

Jeff Nelson

PROJECT: AVOCADO PROJECT

An optimized algorithm for very large sets of parameters given in the Combinatorial Independent Testing (CIT) feature is under active development. When finished, this is expected to be a unique feature that has not yet been generally solved in the industry. At the same time, parallelized test execution support in Avocado is being improved, allowing shorter execution times of the very large test matrices computed by the CIT feature.

The goal of the Avocado project, as a generic test automation framework, is to provide a solid foundation for software projects to build their testing needs on. With Avocado, common testing problems are solved at the framework level, and developers can spend more of their time writing tests that, by default, will adhere to best practices. Avocado has support for different test types that could be written in different programming languages. One of the common problems that Avocado solves at the framework level (and thus makes it available to all test types) is the ability to pass parameters to tests in a uniform way.

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 1:4

COMING IN MAY:

- **Findings from Red Hat Research Day, Europe 2020**
- **Methods for collaborative funding of open research**
- **Breaking cryptographic hardware**

Bringing great research ideas into open source communities

February 2020 – Volume 1: Issue 4

Red Hat