FACILITATING THE SPREAD OF KNOWLEDGE AND INNOVATION IN PROFESSIONAL SOFTWARE DEVELOPMENT

QCon London

SPECIAL REPORT

eMag Issue 40 - March 2016

News, key points, and lessons learned from

QCon London 2016

InfoQ

THANK YOU TO OUR SPONSORS

IBM
Neo4j
AEROSPIKE
APPDYNAMICS
AZUL SYSTEMS

basho
CloudBees The Enterprise Jenkins Company
dynatrace
GridGain

JFrog
PERFORCE
Pivotal
redhat

Solace Systems
Sonatype
ThoughtWorks go snap
WSO2

amazon
BLACKDUCK
EQUAL EXPERTS
eSynergySolutions

GitHub
NGINX
SMARTBEAR

WOLFRAM COMPUTATION MEETS KNOWLEDGE
XebiaLabs Deliver Faster

Media Sponsors

ALPHA GAMMA
DevOps Conferences http://devopsconference.org
Erlang SOLUTIONS
ERLANG
LONDON AJAX

Methods & Tools
stickermule
TRANSYLVANIA JAVA USER GROUP

A LETTER FROM THE EDITOR

Charles Humble

This year was the tenth anniversary for QCon London, and was the first London event produced and run entirely by InfoQ and the C4Media events team after we acquired Trifork's stake in the event last year. It was also our largest London event to date. Including our 140 speakers we had 1,400 team leads, architects, and project managers attending 112 technical sessions across 18 concurrent tracks and 12 in-depth workshops.

Adrian Colyer's opening keynote introduced one of the main themes for this year's event - the importance of computer engineers from industry reading academic papers. A practical example of industry and academia working together followed in Wednesday's keynote when Peter Alvaro and Kolton Andrus described applying Failure Testing Research at Netflix.

Amongst a huge number of other engaging and thought provoking technical sessions were important conversations on topics such as unintended bias in machine learning systems, and the very real dangers of employee burnout.

Attendees had near-instant access to video from almost all of the sessions. We're making these available to everyone as quickly as we can, and have already started publishing them at the rate of five per week. The publishing schedule for presentations can be found on the QCon London web site. You can also see numerous photos of QCon on Flickr.

QCon's focus on practitioner-driven content is reflected in the fact that the program committee that selects the talks and speakers is itself comprised of technical practitioners from the software development community.

QCon London is one of a global series of events run by InfoQ's parent company, C4Media. Hot on the heels of QCon London we held QCon São Paulo on March 28th-April 1st, with QCon Beijing just around the corner starting on April 21st. The next English language QCon is New York starting on June 13th, followed by San Francisco on November 7th.

QCon London will return on 6th-10th March 2017.

This eMag brings together InfoQ's reporting of the event, along with views and opinions shared by attendees.

READ ONLINE ON InfoQ

Continuous Delivery for (Smart) Trucks

by Manuel Pais

Peter Thorngren, Technical Integration/Verification Leader for Volvo Truck Technology Group, spoke at QCon London 2016 on applying continuous delivery techniques to the development of next-generation smart trucks. In particular, virtualization of the hardware components present in a truck, combined with modeling and test automation techniques have been the key enablers for testing new software features in heavy-duty trucks in a matter of minutes as opposed to several months.

Virtual environments allow developers to test their code in a virtual truck and obtain responses similar to a physical truck without having to wait months for integration with hardware. According to Thorngren, a similar hardware based simulator alone would cost in the order of one million euros and allow only serial executions of the test suite.

Modeling the physical world in terms of road and weather conditions as well as the road system itself are crucial for virtualization to provide meaningful results. As Thorngren explained, modeling is never exact but can be kept within a degree of certainty by calibrating it with results from real world tests with actual trucks. The most complex components to model include breaking, engine management and assisted driving systems.

Test automation relies on the virtualization but also on mocking sensors for testing units in isolation. During the presentation Thorngren demonstrated how to manipulate the sensors via a simple GUI on top of the actual virtual sensors used in the tests. A virtual truck is in fact a C++ library using real hardware logic represented by hardware definition files.

The integration process for Volvo's truck software starts with developers running the tests against their local virtual truck instances, then commmiting the code to a build server that runs the same tests with all the changes put together. A green build triggers hardware integration tests in test rigs that replicate parts of a truck's hardware and take around 10 minutes altogether.

In fact, the strategy Thorngren described strongly resembles the inverted test pyramid idea, where costly UI tests are the physical tests using either a real truck or a full-purpose hardware simulator, with all sensors and computing systems interconnected. Because today's heavy-duty trucks possess hundreds of sensors (for comparison a smartphone has between ten and twenty sensors) and dozens of networks - and they keep growing, such tests are very expensive and not scalable. Therefore only a small set is run in real trucks but much more infrequently than the integration and unit tests (which provide sufficient feedback to test new features and quickly find problems).

During his talk Thorngren shared his prediction for a future with autonomous smart trucks, starting with limited self-driving in the next decade, all the way to full fledge self-driving automation in a couple of decades. He also told QCon:

If you look at the truck of the future, you will find more or less a big computer. The truck will likely leverage virtualization (things like VMware or something similar). It will run Java or C/C++. The computational power will be 100's and 100's times bigger than today's cars and trucks. The truck is completely linked with the Internet (as it is today), but it will be more an autonomous vehicle. Exactly when that will happen is obviously a big question, but it is just a matter of time. These trucks will probably be one of the most complex technical things that we see in everyday life. It will be a quite fascinating thing to see.

Throngren stressed how software development techniques like test automation, continuous integration and delivery will be fundamental for quickly adapting vehicles development to new and unpredictable technological evolution. Although Volvo today already upgrades trucks' software via wifi connection (minor changes like modifying a parameter value), Thorngren expects in the future to see automated cloud-like deployment of new virtual machine images to the trucks.

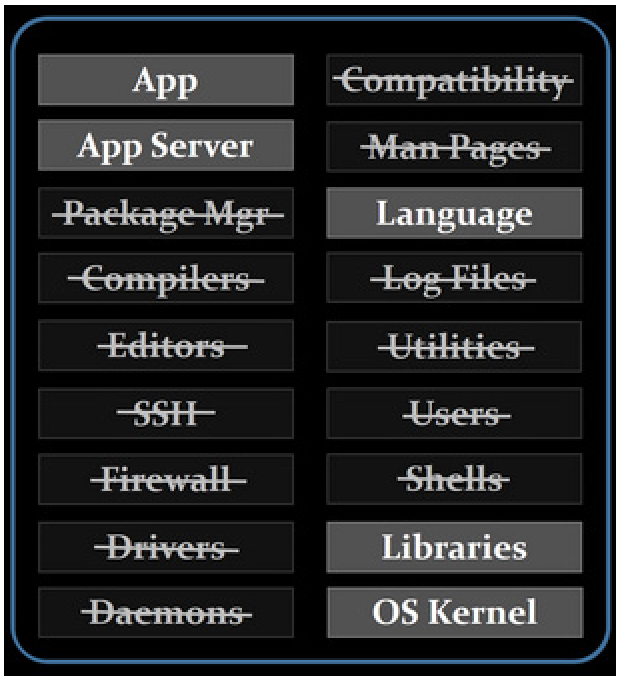
READ ONLINE ON InfoQ

Bootable Apps for Immutable Infrastructure and Security

by Manuel Pais

Axel Fontaine, CEO of Boxfuse, spoke at QCon London 2016 about the "Bootable App" pattern, a bare bones machine image for deploying immutable infrastructure to the cloud. This minimal image covers all layers of the stack, including OS kernel, libraries and runtime environment but still has a small footprint (in the order of megabytes instead of gigabytes), reducing both image upload time and storage costs while also significantly reducing the attack surface on running instances.

Fontaine proposes to include in this minimal image only the strictly necessary components from the lower layers of the stack. The image would then contain the application itself, the application server, the corresponding language runtime and required libraries, and the OS kernel. The figure below shows the typical set of components included in a Bootable App (credit to Axel Fontaine) as opposed to a traditional fully featured image.



Because this minimalist image leaves out many standard OS tools that are common attack points (for instance SSH), the resulting instances bypass both known and future vulnerabilities (Fontaine recommends adopting centralized logging to ensure logs can be accessed long past any instance›s demise).

Another immutability benefit for application security is short instances lifetime as they get replaced on every deployment, thus limiting the duration and extent of a successful attack on any particular instance. Fontaine warns about the implications: instances need to be disposable at any point in time (data stores should have their own security and reliability mechanisms) and sessions should be encrypted and signed in client cookies.

Fontaine also recommends minimizing unnecessary complexity, for cost and security efficiency. Examples include setting up complex service discovery tools when an elastic load balancer might be enough; introducing containers (and the associated image management, scheduling and orchestration, volume management and networking solutions) when the application can run and scale using the native services provided by the cloud platform. Organizations should focus on increasing the business value of their applications, simplifying their delivery process and considering the overall cost of running a service (including time to setup and manage complexity), not just the cloud hosting costs.

READ ONLINE ON InfoQ

Moving from Transactions to Streams to Gain Consistency

by Jan Stenberg

When systems become more complex with each large database split into multiple smaller ones, maybe using derived databases for particular use cases like full text search, a challenge is to keep all this data in sync, Martin Kleppmann stated in his presentation at the recent QCon London conference.

The biggest problem working with many databases is that they are not independent from each other. Pieces of the same data are stored in different forms so when data is updated all databases having a piece of the updated data must also be updated. The most common way of keeping data in sync is to make it a responsibility of application logic, often done by independent writes to each database. This is a fragile solution, in failure scenarios, e.g. after network failure or a server crash, you may fail to update some of the databases and end up with inconsistencies between them. Kleppmann notes that this is not the kind of consistency that eventually will correct itself, at least not until the same data is written again:

This is not eventual consistency; this is more like perpetual inconsistency

The traditional solution is using transactions which gives us atomicity, but Kleppmann notes that this only works within a single database, with two different data stores this is not feasible. Distributed transactions (a.k.a. two-phase commit) can cross multiple storage systems but for Kleppmann they have their own challenges like poor performance and operational problems.

Looking back at the problem, Kleppmann notes that a very simple solution is to order all writes in the system sequentially, and ensuring that everybody then reads them in the same order. He compares to deterministic state machine replication where, with the same starting state, a given input stream will always create the same state transitions when run several times.

With a leader database (the master) where all writes are also stored as a stream in the order they are handled, then one or more follower database can read the stream and apply the writes in exactly the same order. This enables them to update their own data and eventually becoming a consistent copy of the leader. For Kleppmann this is a very fault tolerant solution. Each follower keeps tracks of its position in the stream; after a network failure or crash the follower can just proceed from the last saved position.

Kleppmann mentions Kafka as one tool when implementing the above scenarios. He is currently working on an implementation, Bottled Water, where he is using PostgreSQL to extract data changes which are then relayed to Kafka. The code is available on GitHub.

A presentation about developing with Kafka was recently published on InfoQ.

Kleppmann's presentation is already available for QCon attendees, and will later be available for readers of InfoQ. He has also published the slides from the presentation.

READ ONLINE ON InfoQ

Programming Patterns in Go

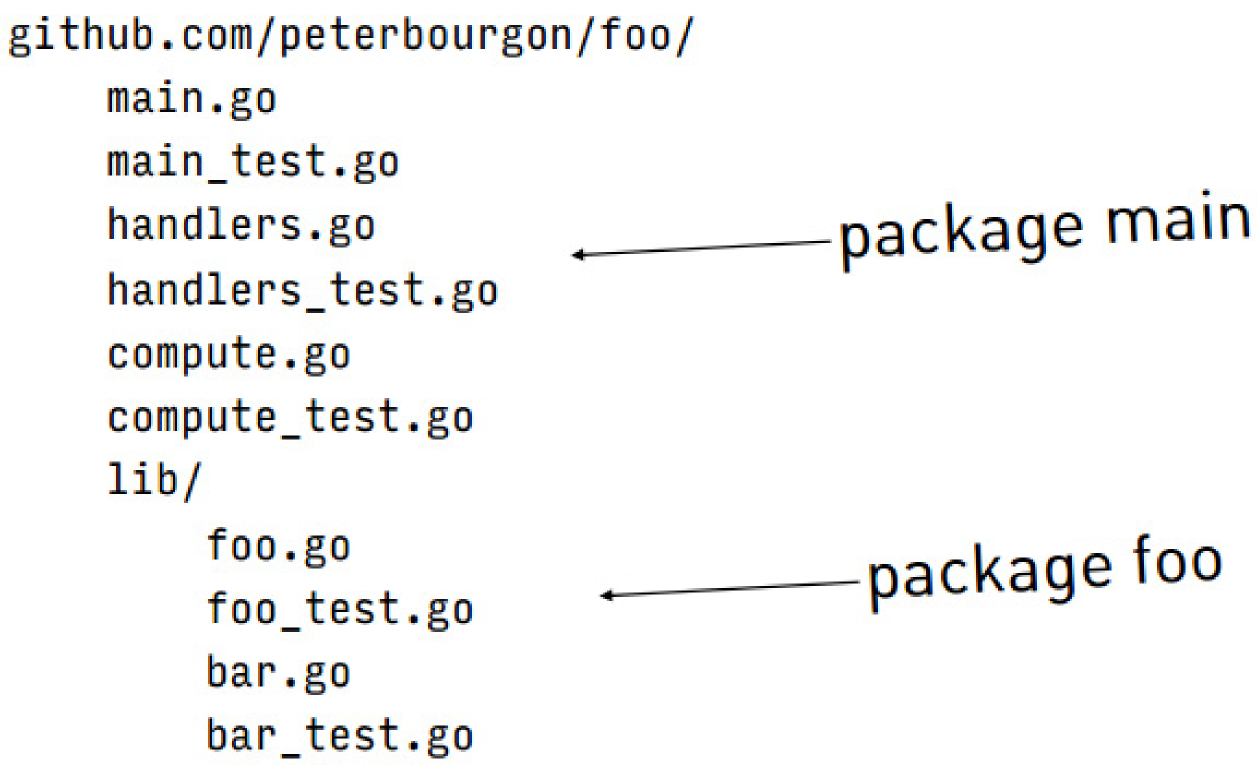
by Abel Avram

Peter Bourgon has recently presented Successful Go Program Design, 6 Years On at QCon London 2016, discussing patterns to use or anti-patterns to avoid when programming in Go. We are covering here in brief many of his suggestions addressed to Go developers.

GOPATH. Put `GOPATH/bin` in the PATH environment variable to make all related binaries easily accessible. Bourgon recommends using a single global `GOPATH`, which works well for most cases. Those who want to make a clear separation between their own code and external dependencies prefer creating two GOPATH entries. There is also the option of building each project separately with gb, without having to set the environment variable.

Repository Structure. The structure of a repository depends on the project. The user may choose any structure (s)he likes if the project is private and will never be made public. If the project is open source, then it should follow the guidance for Remote Packages that can be easily fetched with go get. Bourgon recommends creating a base directory with the main artifacts of the program, and subdirectories for helper packages, as shown in the following image:

Code format. Bourgon stresses the importance of respecting Go's canonical code format style because makes the code more readable once used to it. In his opinion the community treats unformatted code as written by a newbie. gofmt could be run on each save to format the code. In this context he pointed out to Go Code Review guidelines which provide a common language and practices for reviewers and owners. He also supports Andrew Gerrand's recommendations for creating names for variables, functions, exports, etc. in Go, because "everyone will thank you if you do."



```
github.com/peterbourgon/foo/
    main.go
    main_test.go
    handlers.go
    handlers_test.go
    compute.go
    compute_test.go
    lib/
        foo.go
        foo_test.go
        bar.go
        bar_test.go
```

Configuration. Bourgon advises for the configuration settings to be "explicit and well documented." He still uses package flag from the standard library but he wishes it was "less esoteric." He underlines the importance of making the configuration domain explicit. Passing configuration through environment variables does not provide enough information for the user to understand the parametrization of the system, and he considers necessary to include configuration information in the help.

Package Names. Packages should be names based on what they provide rather what they contain. A package containing a `HelloWorld` message should not be called `common` or `consts` but rather `greetings`, to show what it does, not what is contained within.

Dot Import. Bourgon suggests against using "Dot Import", the ability to prefix a package name by a dot enabling the developer to access variables defined in the respective package without mentioning the package name. This makes code less readable for someone new to it, having to figure out what package a variable belongs to. Go "always favors explicitness over implicitness."

Flags. Bourgon does not consider a good idea to initialize flags within the `init()` function but rather the `main()` one because it prevents one to call on the global scope to access something that is a dependency, helping with testing.

Constructors. When calling a constructor, his advice is to inline an initialized `struct` as parameter to avoid passing objects with invalid or incomplete state, as show in the following example:

```go
foo := newFoo(*fooKey, fooConfig{
    Bar:    bar,
    Baz:    baz,
    Period: 100 * time.Millisecond,
})
```

Usable defaults. Instead of initializing a variable of field with `nil`, which requires a `nil` test every time it is used, it is better to initialize the variable with a no-operation value that does nothing. For example, for an `output` variable use `ioutil.Discard`.

Cross-referential components. There is the case when two components contain references to each other. Constructing one of them requires constructing the other one, but the later needs the first one when constructed, as would be the case of the following two `structs`:

```go
type bar struct {
    baz *baz
}
type baz struct {
    bar *bar
}
```

Bourgon presented three variants to deal with this issue:

1. Combine. Two objects that are so related to each other could sometimes be combined into one, in this case a `barbaz` struct.

2. Split. If the two components need to remain separated, then another strategy can be applied depicted in the following snippet:

```
type bar struct {
    a *atom
    monad
}

type baz struct {
    atom
    m *monad
}

a := &atom{...}
m := newMonad(...)

bar := newBar(a, m, ...)
baz := newBaz(a, m, ...)
```

3. Communication. Yet another approach to use when the previous two are not appropriate is to pass messages to each other.

```
type bar struct {
    toBaz chan<- event
}

type baz struct {
    fromBar <-chan event
}

c := make(chan event)
bar := newBar(c, ...)
baz := newBaz(c, ...)
```

Dependencies. One of the top tips shared by Bourgon was "Make dependencies explicit." For example

```
func (f *foo) process() {
    log.Printf("bar: %v", result) //
  ...
}
```

should be replaced with:

```
func (f *foo) process() {
    f.Logger.Printf("bar: %v",
  result) // ...
}
```

`log.Printf` actually calls `Logger`, hiding this dependency. To make it explicit, one needs to create a `Logger` in the construction phase, initialized to `ioutil.Discard` if its value is `nil`.

Channels. Bourgon recommends to use a mutex when sharing memory between goroutines and a channel to orchestrate goroutines.

Logging. Logging can be expensive and can become the bottleneck of an application. As a result, the advise is to log only what is absolutely necessary, information that is read by humans or consumed by machines. Log only info and debug information.

Instrumentation. Bourgon considers instrumentation cheap and recommends using Prometheus to monitor every resource.

Global State. Eliminate implicit global dependencies and global state.

Testing. Perform package testing. Design for testing by writing using a functional style which implies making dependencies explicit as parameters, avoid depending on global state, and use interfaces.

Dependency management. Copy dependencies in the project's repository and use them when building the binaries. Bourgon suggests using gvt, vendetta, glide or gb depending on one's needs.

Build. Use `go install` instead of `go build` because the former caches dependencies and places them in `GOPATH/bin` making them easier to invoke.

These recommendations and others have been applied in creating Go kit, a distributed programming toolkit for building microservices.

Bourgon has used Go from 2009 at SoundCloud and Weaveworks, and has developed Roshi, a time series event database, and Go kit.

The session Successful Go Program Design, 6 Years On presented at QCon London 2016 will be made available for the public later this year.

READ ONLINE ON InfoQ

Microservices for a Streaming World

by Jan Stenberg

Embrace decentralization, build service-based systems and attack the problems that come with distributed state using stream processing tools, Ben Stopford urged in his presentation at the recent QCon London conference

For Stopford, working with Kafka at Confluent, there are many good reasons for building service-based systems. These include loose coupling, bounded contexts, ease of scaling etc., all of which allow us to build systems that can evolve over time. But by taking this approach we are inherently also building distributed systems, which brings its own complexity with issues around latency, failures and so on.

Two fundamental patterns of distributed systems Stopford describes are:

- Request–Response as a way to decouple services, typically using REST, which works well for UIs and when asking questions.
- Event-driven, characterized by asynchronous or "fire and forget" messaging, great for composing complex dependencies across services.

These can also be combined, using request-response for a REST interface and events for background processing.

Looking into asynchronous and event-based communication, e.g. using queues, Stopford sees this as a very simple model, and as long as only one message is pulled at a time the ordering of messages can also be guaranteed. This can scale to a certain degree while still retaining the ordering guarantee, but Stopford notes that at some point we will lose either availability or the ordering guarantee. Another disadvantage he notes is that messages are transient, thus lacking the possibility to go back in time and read old messages after failures.

Stopford believes that a better approach is using a distributed log as a service backbone, with Kafka being one example. Kafka is based on the concept of a Log, which is an append only data structure. This makes both reads and writes efficient, for reads it's a matter of a single seek to a position followed by sequential reads and for writes it's just an append.

Some advantages for microservices that a distributed log enable includes:

- **Always on**, relying on a fault-tolerant broker, like Kafka.
- **Load balancing**, with service instances each reading data from a broker.
- **Fault-tolerant**, since services can fail over but still retain ordering of messages.
- **Rewind and replay**, allowing for a service to return to old messages and replay them, e.g. after an error is discovered and fixed.

One problem not solved is keeping services consistent. After e.g. failures it's hard to avoid duplicate messages ("at least once" delivery) making it necessary for services to be idempotent regarding messages they receive, logically creating an "exactly once" delivery mechanism. Stopford notes that this is not yet available in Kafka (but work is ongoing).

Martin Kleppmann addressed the service consistency problem in his presentation at the same conference.

Stopford's presentation is already available for QCon attendees, and will later be available for readers of InfoQ. He has also published the slides of the presentation.

READ ONLINE ON InfoQ

Using the Actor-model Language Pony for FinTech

by Charles Humble

During his opening Keynote at QCon London on Monday morning Adrian Colyer mentioned the Pony Language:

We're very familiar with the fact that databases and distributed systems have had a lot of influence on each other over the last 5+ years. Now we're starting to see some really interesting work on programming languages that fit into that world. See for example Pony, which came out of Imperial College in London and is really fascinating stuff.

We were fortunate enough to have the designer of the language, Sylvan Clebsch, giving a talk on the native languages track on the Wednesday. Clebsch suggested that Pony is a natural fit for FinTech systems since "...in FinTech we don't write software, we write time-dependent event stream processors that are performance critical but not formally verified". Mostly these are written in Java and C++ though other languages are used including Scala, C, OCaml, Erlang, R and NumPY.

Pony is an actor-model capabilities-secure native language that is compiled ahead of time usingLLVM. The actor model, probably best known from Erlang and, more recently Akka, came from work done by Carl Hewitt and others starting with a paper in 1973. An actor combines state-management with asynchronous methods. In addition to fields, an actor has a single message queue and its own heap. In Pony, Clebsch stated, actor heaps are independently garbage collected and, unlike in Erlang or Akka, the actors themselves are also garbage collected so you don't need to use something like a poison pill message to kill them; in essence there is no manual memory management.

Actors garbage collect their own heaps independently of other actors using a mark-and-don't-sweep algorithm. This means that Pony is O(n) on the reachable graph; unreachable memory has no impact. The actor heap GC has no safepoints, no read or write barriers, no card table marking, and no compacting. Since it doesn't need compaction it has no need for pointer fix-ups.

This means that a pass through a single actor to collect that local heap just means tracing the reachable graph. There is no other associated work of any kind. This means that the amount of Jitter when an actor GCs its own working set is actually quite low. Not only that but it does it before it handles a behaviour.

Pony actors have no blocking constructs. They are cheap, having an overhead of 240 bytes when compared to an object, or 156 bytes on a 32-bit architecture such as ARM. They also have no CPU overhead when they are not executing working code. As Clebsch put it, "If an actor has no work to perform, it's not even in a queue anywhere. The runtime has no knowledge of it, of any kind, unless is has pending work to perform".

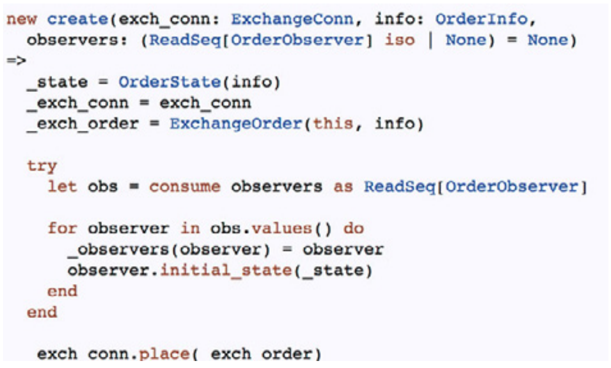
Actors pass messages around using message queues which are intrusive, that is messages do not need to be in more than one queue. More controversially the queues are also unbounded since "if the queue was bounded then, when the queue is full, you have to either block or fail," Clebsch stated. Blocking can introduce deadlocks, whilst failing would require application-specific error handling every time a message is sent. Bounded queues are used to avoid the back pressure problem but, Clebsch argued, whilst unbounded queues move the back pressure problem they don't make it worse. At the time of writing the Pony runtime does not do anything to provide generalised back pressure. Clebsch told InfoQ

That's not the end of the world: it's pretty easy to write domain-specific back pressure, such as the back pressure in TCPListener, which stops accepting new connections when the open connection count exceeds a specified number.

In the next couple of months, generalised back pressure will land in the runtime. What this does is automatically deprioritise actors that send to "loaded queues".

Fundamentally the actor model is about expressing concurrency, and dealing with hard concurrency problems is the main area for which Pony has been designed. Key to that design is the type system which is data-race free, concurrency-aware and proven sound. According to Clebsch there are no other languages with mutability and a data-race free type system, though Rust achieves the same thing through a combination of its type system and atomic reference counting.

Pony has no null. The type system is built on algebraic data types so, in that sense, it can be considered a functional language. The following example, from the talk slide deck, shows some code to create an order on a very basic order management system.



```
new create(exch_conn: ExchangeConn, info: OrderInfo,
  observers: (ReadSeq[OrderObserver] iso | None) = None)
=>
  _state = OrderState(info)
  _exch_conn = exch_conn
  _exch_order = ExchangeOrder(this, info)

  try
    let obs = consume observers as ReadSeq[OrderObserver]

    for observer in obs.values() do
      _observers(observer) = observer
      observer.initial_state(_state)
    end
  end

  exch conn.place( exch order)
```

The `ReadSeq[OrderObserver]` iso introduces us to one of the most important, and novel, concepts in the type system. Iso (Isolated) is a reference capability which offers a guarantee that is built on deny properties. It is these reference capabilities (rcaps) which make the type system data-race free.

"It's not what you are allowed to do, it's what their existence proves cannot exist anywhere else in your program statically. So isolated says it denies both local and global aliases which can either read from or write to the object. That's an incredibly powerful deny guarantee. It means that the most anyone other than you can know about this mutable sequence is its address. They can't read its field or write to its field. That means it is safe to send it to a new actor even though it remains mutable without locks of any kind." Clebsch said.

Rcaps are type annotations that indicate a level of isolation or immutability:
x: Foo iso // An isolated Foo
x: Foo val // A globally immutable Foo
x: Foo ref // A mutable Foo
x: Foo box // A locally immutable Foo (like C++ const)
x: Foo tag // An opaque Foo

It's important to note that data-race freedom using rcaps is handled by the compiler during type checking, which means that there isn't non-linear growth in the amount of compiler work to be done as your code base grows. Colyer provides a fantastic summary of the paper that describes this in more detail on his Morning Paper blog.

Rcaps allow isolated (iso), immutable (val), and opaque (tag) objects to be passed by reference between actors, so you need some way of preventing premature collection of objects in messages (where no actor might have a reference) or where they are reachable by other actors. Pony uses a message protocol for this which is described in a paper which has also been written up by Colyer. The approach is analogous to a consensus algorithm, and Colyer draws parallels with the Chandy-Lamport distributed snapshot algorithm. The Pony paper, "Ownership and Reference Counting Based Garbage Collection in the Actor World" – Clebsch et al. 2015, states

When an actor sends, receives, or drops a reference to an object it does not own, it sends protocol-specific messages to the owner. These protocol-specific messages result in the owner updating its (local) reference count.

Pony is still in very early stages, and some significant items, including reflection and hot code loading, are non-trivial and not yet resolved. That said, whilst a recent survey suggested that the vast majority of users are still just checking out the language, but some are further along. For example Sendence, a NY company, has a FinTech product that they are planning to put into production soon.

Pony is an open-source language and contributions are welcome. There is also a Sandbox so you can try it out yourself.

READ ONLINE ON InfoQ

Anti-Patterns Working with Microservices

by Jan Stenberg

The main problem with monolithic applications is that they are hard to scale, in terms of the application, but more importantly, in terms of the team. The main reason for a switch to microservices should be about teams, Tammer Saleh claimed at the recent QCon London conference when describing common microservices anti-patterns and solutions he has encountered.

With a small application and a small team there is no problem with a monolithic application, but as the application grows and with dozens, or more, of developers they can't all work on the same code base. Moving to microservices enables us to make use of Conway's law in our own favour.

Saleh claims that building with microservices is incredibly complex with a lot of ways that it can go wrong. The most common mistake for him is to just start with microservices. He emphasizes that for all systems you should start as simple as possible and slowly migrate to more complex solutions only when there is need, from a team or business perspective. He notes that the most important thing for a company is to get to the market and make money, not to explore interesting architectural patterns. His solution is instead to start monolithic and extract when necessary, and notes that microservices add a constant tax to development.

With spikes in traffic enough servers must be maintained to handle the peak loads, but with increased traffic and more servers eventually the database will be overloaded instead. One solution to this is to even out the load using queues, basically the queue is a buffer that smooth the traffic out. This makes the communication asynchronous with an increase in complexity since the request-response lifecycle is broken and clients are forced to deal with asynchronous responses. For Saleh this a necessary step though to avoid spending too much money on computing resources.

Services trying to connect to one specific service which is not responding is from an operational perspective a problem. It's hard to do diagnostics and work on the service that is down because of all the requests coming in. With tens or hundreds of other services sending requests this can cause all sorts of problem. A solution is a Circuit breaker which prevents requests reaching the service that is down. When working normally the circuit breaker allows all traffic go through but as soon as it discovers a failure it will prevent further access until the failing service is responding again, at which time the traffic is allowed to pass the circuit breaker again.

With microservices comes a whole new set of testing patterns. One common pattern is mocking the services a service under test is consuming. A team testing a specific service will then instead of communicating with the consuming services write mocks for these services. In the end this means that each team will write their own mocks for all services they consume. This may create a lot of extra work and in Saleh's experience a common solution is to let each team write a common mock for each service they are responsible for, resulting in one mock for each service. An improvement to this that he suggests is that each team writes language specific clients to be used by the consuming services. This client then handles the actual communication with the service and may also include a mock. One advantage Saleh sees in this is that a service has full control over the protocol used and may change it as needed. He also believes this to be a nicer interface.

In a blog post from 2015 Stefan Tilkov argued against always starting with a monolith, instead arguing that for a large enough system we should think about individual subsystems from the beginning.

In a presentation earlier this year Ben Christensen talked about the risk of using a client library, written by the service team, which is the only official way to access a service.

Saleh's presentation is already available for QCon attendees, and will later on be available for readers of InfoQ.

READ ONLINE ON InfoQ

Chaos Testing of Microservices

by Jan Stenberg

The world is naturally chaotic, and we should both plan for and test that our systems can handle this chaos, Rachel Reese claimed at the recent QCon London conference describing how Jet, an e-commerce company launched in July 2015, works with microservices and chaos engineering.

Reese emphasizes how extremely important it is to test the interaction in your environment. Even though all components have been tested it doesn't mean the interactions between them are solid and they can be used together in production, all these have to be tested. She calls Jet a "the right tool for the right job" company, and for her chaos testing is one of the right tools.

Reese defines a microservice as an application of the Single Responsibility Principle (SRP) but at the service level and, because of their functional way of looking at microservices, that it has an input and produces an output. The benefits she sees using microservices include simplified scalability, independent ability to release, and a more even distribution of complexity. Jet runs with somewhere between 400 and 1,000 microservices spread over 10-15 teams, mainly written in F# (a functional-first programming language).

Reese notes that chaos engineering is not about wreaking havoc with the code for fun, instead she defines it as:

Controlled experiments on a distributed system that help to build confidence in the system's ability to tolerate the inevitable failures.

Referring to Principles of Chaos Reese's defines four steps in chaos engineering:

0. Define "normal" (the normal state of the system).

1. Assume "normal" will continue in both a control group and an experimental group.

2. Introduce chaos: servers that crash, hard drives that malfunction, network connections that are severed, etc.

3. Look for a difference in behaviour between the control group and the experimental group.

More specifically this means:

- Build a hypothesis, defining normal behaviour and state of the system like throughput, latency, etc.

- Vary real-world events, spikes in traffic and other things that can make something chaotic.

- Run experiments in production to guarantee authenticity of the tests.

- Automate experiments to run continuously.

The benefits of chaos engineering that Reese has found include:

- Outages occur due to testing during daytime, instead of fixing problems at 3 a.m.

- Engineers start to design for failure.

- It makes systems healthier, by preventing outages happening later on.

Looking at their experiences Reese notes that they are not yet testing in production. As a start-up company their primary objectives has been launching and getting everything right. Right now they are testing in QA randomly at all hours during daytime.

One of their most "interesting" disasters happened a few months ago when their manual testers noticed that their search engine was down, resulting in cascading issues downstream. The reason for this failure was that the chaos testing has restarted the search engine in the wrong way. Due to this single failure they were able to find 5-6 different issues.

Reese concludes by claiming:

If availability matters, you should be testing for it.

Reese's presentation is already available for QCon attendees, and will later be available for readers of InfoQ.

QCon
Amdahl's & Universal Scalability Laws
Speedup
Processors
Amdahl
USL
QCon

Read the full article on InfoQ

As well as InfoQ's own coverage of the event, a large number of attendees blogged and tweeted about their experiences. This is an excerpt from a longer article that rounds up many of those posts.

by Abel Avram

InfoQueue
"ATTEND" A QCON ON A DAILY BASIS BY VISITING INFOQ.COM
1,500,000 DEVELOPERS ACROSS THE GLOBE TRUST US FOR THEIR DAILY DIGEST ON HOT TOPICS FROM THE INDUSTRY.

Domain-driven Design

by Eric Evans

Twitter feedback on this workshop included:

@dthume: All too often, the first decent idea is the last idea - @ericevans0 at #qconlondon

@manupaisable: You don't get all the benefits of modeling just from having a model @ericevans0 #qconlondon

@dthume: There are no nice surprises in software integration - @ericevans0 at #qconlondon

@manupaisable: Duplication inside a context is bad, duplication across contexts can be good tradeoff to allow decoupled evolution - @ericevans0 #qconlondon

@manupaisable: Distinguish core domain from supporting and generic domains, you need to focus energies on the former. @ericevans0 #qconlondon

Scaling Technology and Organizations Together

by Randy Shoup

Twitter feedback on this workshop included:

@heikkih: What if we designed our organizations like we design our systems. Nice group input as start of workshop with @randyshoup at #qconlondon

@heikkih: Really effective organizations are not made up of a small number of large teams, but a large number of small teams @randyshoup #qconlondon

@heikkih: We can engineer the system we want by engineering the organization" @randyshoup follows up on Conway's Law at #qconlondon`

BLT: Babbage, Lovelace, Turing (So, Who Did Invent That Computer?)

by John Graham-Cumming & Sydney Padua

Tim Anderson's attended the keynote:

On Monday evening we got a light-hearted (virtual) look at Babbage's Analytical Engine (1837) which was never built but was interesting as a mechanical computer, and Ada Lovelace's attempts to write code for it, thanks to John Graham-Cumming and illustrator Sydney Padua (author of The Thrilling Adventures of Lovelace and Babbage).

Twitter feedback on this keynote included:

@alblue: "If it's warm, there are going to be cats" — #QConLondon keynote being scientifically as well as historically accurate.

@timanderson: Sydney Padua explaining Babbage's analytical engine #qconlondon "multiplication took 3 mins" https://t.co/Nd4iYd8E1K

Monkeys in Lab Coats: Applying Failure Testing Research @Netflix

by Peter Alvaro & Kolton Andrus

Last day started with a very entertaining and inspiring keynote delivered by Kolton Andrus (Netflix) and Peter Alvaro (University of California). Peter and Kolton shared their experience of a very successful collaboration between industry and academia. Peter had a "big idea", Lineage-driven fault injection, and together with Kolton this evolved from a theoretical model into an automated failure testing system that leverages Netflix's state-of-the-art fault injection and tracing infrastructures.

@palvaro "My code is now actually running live on Netflix…" @KoltonAndrus "…well, minus all of the println statements"

Twitter feedback on this keynote included:

@techiewatt: Collaboration is all about compromise - @KoltonAndrus @palvaro keynote at #qconlondon

@danielbryantuk: Great start to #qconlondon keynote with @palvaro &amp; @KoltonAndrus, with a comparison of success in academia/ industry https://t.co/vSmT6M3dOF

@emirc: Morning keynote, professor and practitioner. 3rd day at #qconlondon https://t.co/vKvC1h2VDJ

@kevj121: Keynote: Break things on purpose :-) to make it better #qconlondon #gemsretail

@danielbryantuk: Popularity contests are fun when you are popular @palvaro on the metrics of success within academia #qconlondon

@danielbryantuk: Yet again a shout out to the benefits of reading academic papers when working within industry by @KoltonAndrus at #qconlondon

@danielbryantuk: Freedom AND responsibility are vital for success within software development. @KoltonAndrus @palvaro # https://t.co/YzLgOdJkE3

@jrubr: Loving how Academia and Industry get together and collaborated to build test failure systems at #netflix #qconlondon https://t.co/8cV6Sk0XRN

@danielbryantuk: Ask "why did a good thing happen?" rather than "could a bad thing happen?" @ palvaro #qconlondon #faultinjection https://t.co/G7ia1eVEe8

@danielbryantuk: HTTP status code 200, but returns an error... Welcome to the real world (it sucks!) @ KoltonAndrus #qconlondon https://t.co/kxe8xk86t6

@KevlinHenney: It's expensive to know everything up front.— @KoltonAndrus #QConLondon

@danielbryantuk: .@palvaro "My code is now actually running live on Netflix..."@KoltonAndrus "...well, minus all of the println statements" #qconlondon

@charleshumble: Adapt the theory to the reality @ palvaro @KoltonAndrus #qconlondon

@robyoung26: "Sometimes it's easier to fit the model to the immutable reality than the opposite." - @palvaro #qconlondon

@oli_codes: The real world is not as "clean" as the purist may like. #qconlondon https://t.co/1BScbnczNF

@techiewatt: Lineage-driven fault injection: @ KoltonAndrus @palvaro #qconlondon: Start with success and work backwards &amp; adapt the theory to reality

@charleshumble: Starting from a successful outcome is a good way to bound a complex problem. @palvarov @KoltonAndrus #qconlondon

Reflections on Software Architecture

by Linda Northrop

Twitter feedback on this keynote included:

@OpenCredo: The quality of software architecture is often the determining factor in the longevity of a system. It’s all about tradeoffs #qconlondon

@techiewatt: architecture - you’ve got one whether you know it or not @LindaNorthrop #qconlondon

@mjbros: Who is the architect of the devops process in an organisation? @lindanorthrup #qconlondon

@charleshumble: What I thought at the time was that the good news we had UML. It was unified. The bad news was we had UML. @lindanorthrop #qconlondon

@ouertani: Must books for architects #qconlondon https://t.co/iJGjaHA9JV

@danielbryantuk: Advancements in software architecture since the 90s. And yes, SOA was an advancement :-) #qconlondon https://t.co/aMNaP4ErZu

@charleshumble: The internet of broken things. Lots of challenges here. @lindanorthrop #qconlondon

@mpaluchowski: #Microservices get us the kind of continuous deployment we want. #qconlondon @lindanorthrop

@danielbryantuk: Evolution of software architecture. The common theme is the ability to continuously deploy new features #qconlondon https://t.co/CzQjwKulY4

@mpaluchowski: Being agile requires #architecture with flexibility designed into the system. #qconlondon @lindanorthrop

@sirgoatofboy: Agility is enabled by architecture, not stifled by it - love it! @lindanorthrop #qconlondon

@ejtweet: Linda Northrop (SEI) at #qconlondon: There is a big difference between being agile and doing agile.

@charleshumble: Bad architecture choices are the top contributor to technical debt. @lindanorthrop #qconlondon

@danielbryantuk: DevOps deployment tools and automation can make software architecture a commodity #qconlondon Slightly controversial, but very interesting

@fabianpiau: Software architecture and refactoring is invisible in the backlog but must be address to minimize the technical debt #qconlondon #fact

@juliotrigo: What colour is your backlog? #qconlondon https://t.co/h7GNyJRuQ8

@charleshumble: Evidence is not the plural of anecdote. @lindanorthrop #qconlondon

@KingPrawnBalls: 4 big architectural challenges (1/2): 1) accelerating capability. New features faster &amp; faster. 2) scale. Planetary scale! #qconlondon

@KingPrawnBalls: 4 big arch challenges (2/2): 3) s/w assurance. Deliver on time in budget on quality. 4) evidence. rely on data not assumptions. #qconlondon

@ludovicianul: Architecture is the medium for doing tradeoff analysis #qconlondon

@charleshumble: Software is *the* building material for our world. #qconlondon @lindanorthrop

@roywasse: You can't manage what you can't monitor - quote from keynote Linda Northrop #qconlondon

@laarchy: 90's - the Golden Age of Software Architecture #qconlondon https://t.co/99qrxQ5sU8

@toughplacetogo: The quality and longevity of a software system is large dependent on the architecture #qconlondon #keynote

@nora_js: If you want scalability, you need to make an architectural decision that supports scalability. - @LindaNorthrop #qconlondon #keynote

Unevenly Distributed

by Adrian Colyer

Luke Bond attended this keynote:

Adrian is the author of The Morning Paper, a daily digest of research paper summaries. Adrian spoke about what he has learned by reading and summarising a paper a day for over a year. He posited that papers A. make us think, B. raise our expectations of what›s possible, C. share applied lessons with us, D.allow us to participate in the "Great Conversation" through the various references cited in the papers we read and E. give us a glimpse into the future (which is already here, but not evenly distributed).

Jeroen Gordijn's attended this keynote:

Adrian had two goals with his presentation. The first goal is to tell the attendees why he is reading so much papers and why the attendees (or you as reader) should read more papers. His second goal is to show some of the fun and amazing research that he has found. Over the course of approximately 1 year he has read an article every weekday and written a summary about every article. By doing this he has read over 350 papers! In his view there are 5 reasons to read articles:

1. **Thinking tools** - By reading articles you sharpen your mind. …
2. **Raise Expectations** - Learn that you don't have to settle for the status quo. …
3. **Applied Lessons** - Articles aren't only written by researchers, but also people who are working in the field. …
4. **The Great Conversation** - If you are really interested in a specific area and you read a lot of papers about the subject you start to see how history builds. There are a lot of references in articles and when you follow these you can see how things changed over time. This will help you place things in context.
5. **Uneven Distribution** - A famous saying goes: "The future is here, it is just not evenly distributed". According to Adrian the here is in research papers. …

Twitter feedback on this keynote included:

@danielbryantuk: I'm hoping to convince developers and architects to read more papers. There are thinking tools, applied lessons @adriancolyer #qconlondon

@eoinwoodz: Five great reasons to read research papers from @adriancolyer at #qconlondon https://t.co/SVhjGMARs8

@dthume: Any system can scale arbitrarily well - if you add enough easily parallelizable inefficiencies @adriancolyer on COST at #qconlondon

@danielbryantuk: The Scalable Communtativity Rule paper providing insight on good API design... with @adriancolyer #qconlondon https://t.co/wl74UpCgCH

@feroGarcia: Do less testing...but don't sacrifice quality...! Awesome paper from the @Microsoft team #qconlondon

@charleshumble: You can save millions of dollars if you do less testing. @adriancolyer #qconlondon

@mpaluchowski: Computer used to be a job title. People went to work and did computing. @adriancolyer #qconlondon

@alblue: Lots of stuff coming up in computing — @adriancolyer at #QConLondon https://t.co/uC9LOZHgnp

@alblue: The new latency numbers everyone should know — @adriancolyer at #QConLondon https://t.co/ROFjhxlj7Q

@jovianjake: Highly encouraged to read papers by @adriancolyer at #qconlondon . Now that's a surprise

@KingPrawnBalls: #qconlondon keynote: share something you've learned every day and it soon compounds into a great resource. @adriancolyer

Ending the Chain-of-blame: Continuous Consequence

by Katherine Kirk

Magnus Ljadas' attended this session:

Katherine Kirk concluded, the 3 characteristics for human existence is also true for our business:

- Everything is in a constant state of change
- We need to collaborate
- We will always battle dissatisfaction

Twitter feedback on this session included:

@ocklund: @kkirk Be grateful if it's your fault because then you can change it. If you pass on blame you make yourself helpless #qconlondon

@merybere: You don't loose your intelligence because you said I appreciate you Katherine Kirk talking about gratitude #qconlondon

Far from the Mobbing Crowd

by Steve Tooke & Matt Wynne

Twitter feedback on this session included:

@kevj121: Issues and concerns resolved faster and remote working all together #qconlondon #gemsretail https://t.co/Xl4K6C5qnc

@roywasse: Learning about #Mobb programming. It's about the teaming, not about being in a team. #qconlondon

@KingPrawnBalls: Effective remote team: foster a sense of adventure, community, purpose and play. But allow independence. #qconlondon https://t.co/HZszs1PgXF

How to Win Hearts and Minds

by Kate Gray & Chris Young

Twitter feedback on this session included:

@paulacwalter: The only thing you know about people is that they all have different agendas. @grisgraygrau #qconlondon

@danielbryantuk: Fantastic insight into driving tech cultural change with methods from electoral politics by @grisgraygrau and @worldofchris #qconlondon

@danielbryantuk: Segmentation, vision and polling - vital for driving a successful campaign! #qconlondon @grisgraygrau and @worldofchris

@amertner: If you want people to do something different, persuade them through reason and convince them through emotion #emotion #qconlondon

@paulacwalter: To deliver effective business change, be very clear about the goal and how you articulate it. Relevant &amp; meaningful. #qconlondon

@paulacwalter: Do the hard work to find out why people are not with you right now. Show respect for others. #qconlondon @grisgraygrau

@willhillbet: Love Failure & Embrace the Fall Out

by Gavin Stevenson

Tim Anderson's attended this session:

William Hill's R&D Engineering Lead Gavin Stevenson told attendees that they should celebrate IT failures. "The best time to understand how your system works is when it is dying," he said.

Stevenson's underlying point is that examining how an application fails when under stress is more illuminating than simply observing it working. Failure identifies the limitations of the system.

Stevenson's team also relies on Docker containers for deployment. "Everything we do in R&D, it's Docker," he said; though they have struggled with container load-balancing and orchestration. "There isn't a brilliant solution," he said, though they are looking at Docker Swarm, a product for clustering Docker engines.

"It's a reactive microservice-based architecture," said Stevenson. "Probably. Nobody seems to agree what microservices is."

Peter Liljenberg attended this session:

Gavin Stevenson … talked about WilliamHills betting engine and how they are transitioning from a large database centric solution to a microservice-based architecture (this was a common theme during the conference). They were building a "production ready" betting engine in 2 week sprints, testing it with real production data. The most interesting takeaway was how important it is to really try to break your system. When the system fails, that's when you learn. The old saying "If it ain't broken, don't fix it" just doesn't apply anymore.

"If it ain't broken, try harder!" – Gavin on testing

Twitter feedback on this session included:

@danielbryantuk: Data flows through the system. We try to avoid explicit pushing or pulling. This improves fault tolerance @WillHillBet #qconlondon

@stefanoric: William Hill new settlement engine is written with Erlang #qconlondon

@danielbryantuk: Erlang syntax may initially make your eyes bleed, but use it for a couple of weeks and you'll like it. #qconlondon

@OpenCredo: The supervision model and pattern matching of Erlang is very useful for building robust and scalable solutions @willhillbet #qconlondon

@IzidorMatusov: If it isn't broken, break it and learn from the failure. #qconlondon

@timanderson: Which programming language? is the wrong question says Gavin Stevenson #qconlondon. Find good people, recruitment is hard.

@paulacwalter: small applications, good people and operational support are what matters, not the language you choose #qconlondon

@pliljenberg: If it ain't broken, try harder - Gavin Stevenson @WillHillBet talking about embracing failure. #qconlondon

@kylethompson86: First talk of the day at #qconlondon by william hill emphasizing the importance of monitoring because your tests will NOT cover everything

Cassandra at Apple Scale

by Sankalp Kohli

Alex Blewitt attended this session:

There was also a Cassandra at Apple Scale presentation, including some of the failure modes that they have seen. These have resulted in a number of fixes being applied to the codebase, including signatures for host membership and incremental rebuilding to avoid problems when incomplete data and group membership change at the same time.

Twitter feedback on this session included:

@andyhedges: Apple has 100,000 instances of cassandra in production #qconlondon @ kohlisankalp

@v3rtti: We have a lot of data. And it's real data, not analytics data. @kohlisankalp #lol #qconlondon

Staying in Sync: From Transactions to Streams

by Martin Kleppmann

Peter Liljenberg attended this session:

Martin Kleppmann held one of the best presentations of the whole conference where he talked about keeping data sources in sync, moving away from (distributed) transactions to streams. The content was not very in-depth, but Martin had deep knowledge of the subject, excellent slide and a lot of energy when presenting. This is a talk everyone should watch and learn from.

“Stupidly simple solutions are the best”
– Martin Kleppman

Magnus Ljadas’ attended this session:

Martin Kleppmann … proposed a stupid simple solution, as he put it, as an alternative to distributed transactions by using a distributed commit log.

Twitter feedback on this session included:

@mpaluchowski: Synchronization is probably a 40-year-old problem and embarrasingly we still don’t have a good way of handling it. #qconlondon @martinkl

@mpaluchowski: Keeping systems in sync is essy when everything goes well. Handling failure is hard. #qconlondon @martinkl

@mpaluchowski: It’s not eventual consistency. Won’t resolve itself. It’s perpetual inconsistency. #qconlondon @martinkl

@mpaluchowski: Most synchronization problems are order problems. Fixing order wins half the battle. #qconlondon @martinkl

@Solisolitude: Stupidly simple solutions are the best #qconlondon

@ocklund: @martinkl makes a strong case using ordered log (Apache Kafka) instead of transactions to stay in sync #qconlondon https://t.co/VIZoEG6tdZ

@MrAndyButcher: Event streams + total ordering, to avoid “perpetual inconsistency” and deadlock. Excellent talk from @martinkl as always #qconlondon

#NetflixEverywhere Global Architecture

by Josh Evans

Peter Liljenberg attended this session:

Josh Evans from Netflix presented how Netflix have expanded there streaming services to almost the entire globe (Netflix#Everywhere). Netflix have had some major outages and failures, both in their own software and the underlying cloud AWS-platform. Josh concludes that "Failure is inevitable" and that one really have to embrace the failure and not fail in the same way twice. This had lead Netflix to embrace a "Failure-driven architecture" approach when building their platform.

Josh presented Netflix four architecture pillars: data, caching, traffic and microservices, and how they use (among other techs) EVCache, Cassandra and DNS to keep their services up and running in case of total failures of an AWS datacenter/region. He also showed how they test failure in different regions and route traffic to another region to minimize customer impact. If infrastructure and architecture at scale is of any interest, watch this talk when it comes online!

"Never fail in the same way twice" – Josh Evans

Twitter feedback on this session included:

@worldofchris: Failure is inevitable, finger pointing doesn't help. Learn and never fail the same way twice @Ops_Engineering @netflix #qconlondon

@robyoung26: Soft routing gave Netflix the tool they needed to evacuate a large-scale regional outage. #qconlondon

@robyoung26: Visualizing error rates at @Netflix. Soft routing and then DNS-based region isolation during incident #qconlondon https://t.co/hXcB8y2pQ4

@robyoung26: The time-series anatomy of a @Netflix regional failure exercise #qconlondon https://t.co/MF9vBcMNd6

@KingPrawnBalls: Think globally, act locally. Netflix arch solutions were always taking them a step further towards going global. #qconlondon /Josh Evans

@robyoung26: Availability at @Netflix increased once they got more rigorous with their failure rehearsal exercises #qconlondon https://t.co/Anc6vBOgg6

Architecting Google Docs

by Micah Lemonik

Tim Anderson's attended the session:

Lemonik was involved in a small company called 2Web technologies which developed an Excel-like engine in 2003-4, and joined Google (which acquired 2Web) in 2005 to work on Google Sheets. The big story here was the how Google Sheets became collaborative, so more than one person could work on a spreadsheet simultaneously. "Google didn't like it initially," said Lemonik. "They thought it was too weird." The team persisted though, thinking about the editing process as "messages being transferred between collaborators" rather than as file updates; and it worked.

You can actually use today's version in your own projects, with Google's Realtime API, provided that you are happy to have your stuff on Google Drive.

I particularly enjoyed Lemonik's question to the audience. Two people are working on a sheet, and one types "6" into a cell. Then the same person overtypes this with "7". Then the collaborator overtypes the same cell with "8". Next, the first person presses Ctrl-z for undo. What should be the result?

The audience split neatly into "6", "7", and just a few "8" (the rationale for "8" is that undo should only undo your own changes and not touch those made by others).

Google, incidentally, settled on "6", maintaining a separate undo stack for each user. But there is no right answer.

Lemonik also discussed the problem of consistency when there are large numbers of contributors. A hard problem. "There have to be bounds to the system in order for it to perform well," he said. "The biggest takeaway for me in building the system is that you just can't have it all. All of engineering is this trade-off."

Alex Blewitt attended this session:

Apparently Google Docs started off as a web-based excel spreadsheet server; a small company called 2WebTechnologies had a product called XL2Web that provided a remote viewing platform for spreadsheet content, by interpreting an Excel spreadsheet on the server and then rendering a remote HTML view. Since the leading browser was IE SP1 at the time, the data had to be stored on the server in order to came through. The collaboration was an accident as two people edited a document at the same time and they both saw the results. Fortunately the model they built had no non-commutative operations which meant that operations could be replayed. It's the same model in Google Docs SDK today; and the fact that the APIs existed helped mobile adoption later on. Scaling is through sharding and consistency trade-offs; for a popular document in read/write mode, users may be switched to a read-only version that may be delayed, thereby trading availability for consistency. It turns out that once the unit of consistency is too large for a single server, it's no longer consistent, which means finer-grained control of document sharding often implies sharding at a greater level, like per chapter on a book.

Twitter feedback on this session included:

@alblue: "If Google asks you something as a startup, you say 'yes'" - on moving calculation to browser #QConLondon https://t.co/QVgVcdHcPU

@alblue: "The leading browser in the time was IE6 SP1 — let that sink in for a minute" #QConLondon https://t.co/5kDmu3nYGZ

@alblue: "Storing data server side was an artifact of how bad the client (IE) was" #QConLondon https://t.co/ZH1tyQu3Bd

@alblue: "The collaboration came about by accident—the first time we saw two people editing documents at the same time we were intrigued" #QConLondon

@alblue: "A feature which came about by accident—snapshot function was used to reconnect clients but became undo" #QConLondon https://t.co/oEwCosr7RF

@alblue: "The spreadsheet model we happened to build had no non commutative operations" - describing a happy accident #QConLondon

@alblue: "The transformations for this kind of model are still used today in Google Docs SDK" #QConLondon https://t.co/ziyfBS0AsR

@alblue: "The users love it — the engineers hate it. That's how you know you've arrived at the right solution." #QConLondon

@alblue: "We were lucky when phones and API clients showed up—because shared objects could immediately deliver" #QConLondon https://t.co/YoNw7VM7CZ

@alblue: "Once your unit of consistency is too large for a single server it's no longer a unit of consistency" #QConLondon https://t.co/bMFU1UfHDH

@alblue: "Let me just check my notes on what I can say about Google storage … ...Google storage has a lot of data" #QConLondon

QCon
THE WORLD'S FIRST REGULAR HIGH DEFINITION TELEVISION SERVICE WAS INAUGURATED HERE BY THE BBC 2 November 1936

Cloud-based Microservices Powering BBC iPlayer

by Stephen Godwin

Tim Anderson's attended the session:

Stephen Godwin spoke on Microservices powering BBC iPlayer. This was a compelling talk for several reasons. The BBC is hooked on AWS (Amazon Web Services) apparently and stores 21TB daily into S3 (Simple Storage Service). This includes safety copies. iPlayer was rebuilt in 2013, Godwin told us, and the team of 25 developers achieves 34 live deployments per week on average; clearly the DevOps stuff is working here. Godwin advocates genuinely "micro" services. "How big should a microservice be? For us, about 600 Java statements," he said.

Twitter feedback on this session included:

@KingPrawnBalls: 10,000 hours of media published in an average week on BBC iPlayer. 10m requests to play video each typical day. Video @ scale! #qconlondon

@timanderson: BBC iPlayer was rebuilt almost from scratch in 2013 apparently #qconlondon

@timanderson: BBC stores 21TB per day into Amazon's S3 storage service #qconlondon

@danielbryantuk: Learnings from the BBC's use of AWS S3 SDK were fed back to Amazon, and changes were made (TCP timeout option on conn re-use) #qconlondon

@danielbryantuk: I'm not going to say how big microservices should be, but at the BBC we have converged on about 600 lines of Java @ SteveGodwin #qconlondon

@worldofchris: Cool. @BBCiPlayer run @netflix #chaosmonkey in production in @awscloud @ SteveGodwin #qconlondon

@mpaluchowski: An empty queue is a happy queue - is our rule. #qconlondon @SteveGodwin

@Mollydogsdad: Bbc can deploy iplayer to live in 15 mins. Impressive. #qconlondon

@timanderson: BBC iPlayer team: 25 devs, 34 live deployments per week. "Devs should spend 60% of their time writing tests" #qconlondon

@danielbryantuk: We write outside-in tests with Ruby, even though we are a Java shop. No code sharing forces use of 'front door' @SteveGodwin #qconlondon

@mpaluchowski: You can tell which services are too big. That's the ones developers don't want to work with. #qconlondon @SteveGodwin

@danielbryantuk: Elastic scaling is good, but linear scaling is excellent. This helps capacity planning, resourcing, and costing @SteveGodwin #qconlondon

ECS & Docker: Secure Async Execution @Coursera

by Brennan Saeta

Twitter feedback on this session included:

@fabianpiau: Back in 2012, @Coursera was a PHP monolith app! #qconlondon

@parcelbizerba: @bsaeta from coursera talks about their journey from PHP monolith to scala microservices #qconlondon https://t.co/wVBV4gskyQ

@fabianpiau: @coursera has developed its own job scheduler called #iguazu, a layer in between Amazon EC2 to manage autoscaling #qconlondon

@fabianpiau: @coursera has huge security challenges! Which company can say it is running arbitrary code on its own server? #codingAssignement #qconlondon

@fabianpiau: To face this security challenge, @coursera is using containers (@Docker) #previoustweet #qconlondon

@WelshSeanSter: How would you like random people from the Internet to upload code and then you compile and run on your infra? @bsaeta #qconlondon @coursera

Hot Code Is Faster Code - Addressing JVM Warm-up

by Mark Price

Twitter feedback on this session included:

@charleshumble: We get about a 20x speed up on trivial methods between interpreted and compiled #java byte code. #qconlondon

@charleshumble: JVM warm-up strategies: Warm-up in-place - send data through the system. Must be production like. #qconlondon @epickrram

@charleshumble: jitwatch - Log analyser / visualiser for Java HotSpot JIT compiler. Inspect inlining decisions, hot methods, etc. @epickrram #qconlondon

@charleshumble: JMH profiler - can only use in the context of a JMH benchmark but provides deep inspection of code. @epickrram #qconlondon

Java 9 - the (G1) GC Awakens!

by Monica Beckwith

Twitter feedback on this session included:

@charleshumble: In JDK9 the Heap Occupancy threshold for G1 is adaptive. Currently defaulted to 45%. #qconlondon @mon_beck

@mjpt777: Monica @mon_beck Doing a great job of explaining G1 GC but wow is it ever complicated. #qconlondon

@charleshumble: A heavily tuned JVM command line may be restricting the G1 GC ergonomics and adaptability. @mon_beck #qconlondon

Netty @Apple: Large Scale Deployment/ Connectivity

by Norman Maurer

Alex Blewitt attended this session:

Norman Maurer talked about Apple's use of large-scale Netty deployments (over ˝ million), and some of the contributions that he and others at Apple had made to the framework. One of them was a high-performance networking and SSL termination layer, originally ported from Twitter's Finagle. Part of the problem is that Java's NIO has too many synchronized locks and the Java SSL libraries generates too much garbage, which prevents maximizing core usage. By providing a different SSLProvider for Netty and binding to OpenSSL/ LibreSSL/BoringSSL he was able to demonstrate an increase from 10Mb/s to 60Mb/s and from a 40% utilization of a multi-core box to 100% utilization for a load tester.

Twitter feedback on this session included:

@alblue: Problems with NIO:* Selector. selectedKeys() produces too much garbage* Synchronized everywhere* Internal copying of buffers #QConLondon

@dthume: ...Although they can't tell us *which* services are netty based :-) #qconlondon

@dthume: 550,000 netty instances in production at Apple. 'Nuff said. #qconlondon

@alblue: "JNI has an awesome API" – sarcasm is live and well at #QConLondon

@alblue: Using Epoll transport for Netty results in less GC pressure and uses advanced socket setup #QConLondon https://t.co/eeMmO9doYH

@charleshumble: Who likes using ByteBuffers? No hands go up. #qconlondon @normanmaurer

@charleshumble: JDKs SSL implementation is very slow. With a naive benchmark 16mb/second, unable to utilise all cores. @normanmaurer #qconlondon

@charleshumble: JNI based SSLEngine was part of @twitter Finagle, now part of Netty. 4x faster than the JDK implementation. #qconlondon

@andyhedges: ~100k TPS with JDK SSL, then ~500k TPS with netty equivalent on same box. Netty fully uses the server's CPU resources too. #qconlondon

@alblue: Asynchronous IO frameworks need to take account of back pressure to avoid unconstrained memory problems @normanmaurer at #QConLondon

@pliljenberg: Massive scale? 10s PetaBytes of data through #netty every day at Apple - @normanmaurer #qconlondon

@dthume: Atomic*fieldupdater saved 4gb of heap w/1billion connections in netty #qconlondon

Project Jigsaw in JDK 9: Modularity Comes to Java

by Simon Ritter

Twitter feedback on this session included:

@fabianpiau: To date 23 classes, 18 interfaces and 379 method have been deprecated, nothing removed #javafacts #qconlondon

@fabianpiau: #JDK9 has a new directory tree, bye bye the jre folder and its lib folder containing some duplicated files! #qconlondon

@fabianpiau: From JDK8, the command line tool jdeps has been introduced to understand the static dependencies of your apps and libraries #qconlondon

@stealthness: #qconlondon Simon Ritter - sun.misc.Unsafe such a hidden gem to be lost in modularity. https://t.co/AQIWkhp4Me

@charleshumble: "We should definitely, hopefully, get Jigsaw in JDK 9. No promises though." @speakjava #qconlondon

@fabianpiau: With #jigsaw, the 'public' keyword now has different flavours depending on the module visibility and configuration #qconlondon

@charleshumble: No more classpath hell. Now we have -modulepath hell. It is better though. @speakjava #qconlondon

Configuration Classes with Autowired Constructors

```
@Configuration
public class MyBookAdminConfig {

    private final DataSource bookAdminDataSource;

    // @Autowired
    public MyBookAdminService(DataSource bookAdminDataSource) {
        this.bookAdminDataSource = bookAdminDataSource;
    }

    @Bean
    public BookAdminService myBookAdminService() {
        MyBookAdminService service = new MyBookAdminS          ();
        service.setDataSource(this.bookAdminDataSourc
        return service;
    }
}
```

Spring Framework 5 - Preview & Roadmap

by Juergen Hoeller

Twitter feedback on this session included:

@charleshumble: Source code readability is the heart of what we're doing with our programming model. @springjuergen #qconlondon

@fabianpiau: In #spring 5, the autowired annotation becomes optional #qconlondon

@charleshumble: Themes for Spring 5.0: JDK 9, HTTP/2 Reactive architecture. Java 8+ baseline, skipping Java 7, #JUnit 5. @springjuergen #qconlondon

@charleshumble: It looks like Jigsaw will have no concept of versioning, just structuring + visibility enforcement. @springjuergen #qconlondon

@charleshumble: Servlet 4.0 is an excuse for servlet vendors to not support HTTP/2. HTTP 1.1 is 20 years old. #qconlondon https://t.co/oalN9q73Bz

@ludovicianul: Spring Boot 1.4 will make HTTP2 a first class citizen #qconlondon

@charleshumble: Spring reactive is a Spring MVC like endpoint model. It is being developed as a public R&D project - https://t.co/vmJT9fjzsC #qconlondon

How Will Persistent Memory Change Software Design?

by Maciej Maciejewski

Magnus Ljadas' attended this session:

New innovations in hardware will impact how we build systems: persistent RAM will fundamentally change how we database stuff, and hosts being able to access RAM on other hosts over super low latency network channels without consuming CPU will impact what's possible to build.

The Quest for Low-latency with Concurrent Java

by Martin Thompson

Alex Blewitt attended this session:

His open-source Aeron message server uses an out-of-process message broker that runs on the client and delivers messages outside of a Java process, so that client-side Java garbage collections don't introduce unnecessary jitter due to GC pauses. Messages are passed by appending into a rotating set of queues and then shared memory is used to allow the client to write directly into the memory space read by the server. He also compared the standard Queue objects in Java, and showed that they either generated garbage, had locks, or both, and that the latencies (particularly the 99th percentile) were significant once contention between multiple producers kicked in. He then introduced the ManyToOneConcurrentArrayQueue and showed benchmarks to demonstrate its effectiveness over other standard queue mechanisms.

Twitter feedback on this session included:

@alblue: "If a system does not respond in a timely manner then it is effectively unavailable" – @mjpt777 at #QConLondon

@alblue: The performance benchmarks mentioned by @mjpt777 at #QConLondon talk are at @github here:https://t.co/tKEBzBUytj

@silverSpoon: Logging is a messaging problem @mjpt777 #qconlondon

@santmatthew: @mjpt777 is so right: If you don't know Amdahl's law and queueing theory, they will hunt you down. #qconlondon

Understanding Hardware Transactional Memory

by Gil Tene

Alex Blewitt attended this session:

Although HTM was attempted in Haswell, it was disabled due to firmware flaws – but now the latest generation of Broadwell chips have HTM enabled through the TSX instruction. This allows a transaction begin to occur (with a call-out to a cleanup/retry location if it fails) and then all cache state modified during the subsequent instructions stays in the cache until the commit, at which point the modified changes are written back or the state is restored as at the beginning of the transaction and the retry logic is called. This allows for some specific improvements (particularly with lock elision using effectively free optimistic locking) and particularly if the locks in case are over a variety of different data structures the speculation can provide increased concurrent throughput.

Peter Liljenberg attended this session:

Gil Tene talked about Hardware Transactional Memory. Really low-level stuff about CPU pipeline and cache optimization. HTM in the JVM is not new, Azul has been delivering both hardware and a customized JVM with JVM for 10 years. What's interesting is that it will become mainstream now when Intel is shipping CPUs with support for HTM. Gil succeeded in a very educational way describe the complexity of HTM and how it can be implemented in for example the standard JVM. In the end Gil talked about how the developers must reason about locking and synchronization to make the most of HTM in their code.

Build, Ship and Run Unikernels

by Justin Cormack

Twitter feedback on this session included:

@mpaluchowski: There's code you want to run and the code your OS includes, that comes along for the ride. #qconlondon @justincormack

@fintanr: Operating systems are just kind of technical debt -@justincormack #qconlondon &lt;&lt; at times it's hard to disagree here

@mpaluchowski: That Unix code written in the 70s is still there. Lots of technical debt in system programming. #qconlondon @justincormack

Observe, Enhance, & Control: VMs to Containers

by Mitchell Hashimoto

Peter Liljenberg attended this session:

Mitchell argues that the "state-of-the-art" tools from the age of VMs are not really suited to handle the tasks anymore. Even though the tools are extremely good, they do solve a completely different problem.

Twitter feedback on this session included:

@fintanr: Observe, enhance, control @mitchellh #qconlondon https://t.co/sv4mK7u0y1

@fabianpiau: There was the age of virtual machines back in 2006, now we are in the age of containers @ mitchellh #qconlondon

@toughplacetogo: "Nowadays banks are software companies that happen to do finance" #qconlondon #containers @mitchellh

@fabianpiau: @mitchellh "If your software is not API driven, then it is probably a second choice one..." Thanks god, I'm working on a API! #qconlondon

@fintanr: 2016 datacenter problems - infrastructure management, service discovery, configuration and scale: speed and size @mitchellh #qconlondon

@fintanr: Updating a server and waiting for an hour for dns to sync is just not an option - @mitchellh #qconlondon

@fabianpiau: Configuration is the main problem when working with containers and microservices. Couldn't agree more! #qconlondon

@fintanr: The world in 2016 from @mitchellh #qconlondon https://t.co/l808DhY64Q

@fintanr: key value stores as replacement for traditional config management tools for #Containers - @mitchellh #qconlondon

Natural Language Processing (NLP): Here Be Dragons

by Emma Deraze

Twitter feedback on this session included:

@charleshumble: Any problem you possibly have with txt you can have with @twitter Emma Derate, #qconlondon - for e.g. Hashtag wars

@charleshumble: Sentiment analysis. How do people feel about this. I don't know and honestly I don't care. Emma Derate #qconlondon

@charleshumble: The only thing sentiment analysis is good for is reviews, but you have the star rating so you already know Emma Deraze #qconlondon

@charleshumble: Bosch sues Dyson over vacuum cleaner claims. Is that positive or negative. I don't know, it depends. Emma Derate #qconlondon

@megamda: NLP is all about high probability and speculation != accuracy #qconlondon #nlp #machinelearning

Continuous Delivery: Benefits Explained

by Lianping Chen

Twitter feedback on this session included:

@amertner: Cycle time improvements have much larger impact than productivity improvements. #cd #qconlondon

@fiddur: Continuous Deployment becomes so boring so we deploy at peak time to make it interresting. #qconlondon

@phuturespace: #qconlondon. LianPing Chen's Paddy Power success story for Continuous Delivery. More releases per month. No releases per weekend.

Immutable Infrastructure: Rise of Machine Images

by Axel Fontaine

Twitter feedback on this session included:

@chrismare: #qconlondon @axelfontaine treat servers like cattle and not pets

@andyhedges: CRUD for servers is dead. We are discarding the U @axelfontaine #qconlondon

@roywasse: Complexity is the enemy of security #qconlondon (quote from @axelfontaine immutable infrastructure talk)

Hacking Bank Mobile Apps

by Stevie Graham

Tim Anderson's attended the session:

… He set himself the task of analyzing mobile apps from banks like RBS and Barclays to discover how they communicated with their servers….

One idea is a man-in-the-middle proxy, where the app communicates with your server thinking that it is talking to the bank's servers, but this does not work with banking apps, he explained. They use a technique called SSL pinning, where the app has a copy of the bank's security certificate, and verify that the server is using that same certificate for encryption.

There are ways round this, Graham remarked, but it is not worth trying to circumvent it. Instead, he used a technique called method hooking, where the functions in the app itself are augmented by the hacker. Objective-C makes this easy, he said, because of its dynamic dispatch system which defers the decision about which function to call until runtime. "You can insert shims that decorate or completely replace implementations," he said.

It is not as simple as that though, thanks to steps taken by the banking app developers to make them harder to reverse-engineer.

"Some banks, like Barclays, take numerous steps to obfuscate," Graham said. Nor did he share all his secrets, but he was able to demonstrate his success; though he said the effort took him most of a year….

"I have possession of the device. I can see the data that's on the device. If the OS can execute the app, it's possible for a human theoretically to reason about how it works. Barclays maybe want to protect their users from malware. That's a legitimate reason for hardening the apps. I'll still crack them. It's going to be an arms race, a game of whack-a-mole," said Graham.

"I don't want to reverse-engineer apps. I did it as a conversation starter, here's what's possible," he adds. He has, perhaps naively, a belief in the democratizing power of technology to make life better for ordinary people.

Twitter feedback on this session included:

@timanderson: Oauth - "that's another talk on why it's not secure enough" #qconlondon

@timanderson: Question to speaker: "any risk of legal issues"? "We'll find out, stay tuned" #qconlondon

@timanderson: I'm on your side, but a lot of people above me are not says banking IT guy in audience #qconlondon

Building Trust Machines using the Block Chain

by Ken Kappler

Twitter feedback on this session included:

@eoinwoodz: BlockChain is about consensus across a distributed network, not just Bitcoin @KapplerKen #qconlondon

@eoinwoodz: BlockChain doing for financial transactions what watches did for time (ubiquitous visibility and consensus). @KapplerKen #qconlondon

Fighting the #Fintech Wave with DevOps

by Benjamin Wootton

Twitter feedback on this session included:

@chrismare: #qconlondon @axelfontaine treat servers like cattle and not pets

@andyhedges: CRUD for servers is dead. We are discarding the U @axelfontaine #qconlondon

@roywasse: Complexity is the enemy of security #qconlondon (quote from @axelfontaine immutable infrastructure talk)

Meet the Node.js Anti-patterns

by Pedro Teixeira & Igor Soarez

Luke Bond attended this session:

YLD's co-founder and CTO Pedro Teixeira was joined by fellow YLDer Igor Soarez in a Node.js double-act discussing anti-patterns and bad practices in Node.js development. Built around a narrative of Jane, a Java developer doing her first Node.js project, we learned about callback hell, code style, error handling and some architecture and scalability challenges. Packed full of lessons, this will be one of those videos you go back and watch many times to extract as much as possible from it.

Twitter feedback on this session included:

@matthewrevell: Callback hell is the first #nodejs antipattern Pedro and Igor cover at #qconlondon https://t.co/9dV90SFg0f

@lukeb0nd: Solutions to callback hell in #nodejs @ pgte @igorsoarez @YLDio #qconlondon https://t.co/Rl073r0CGr

@stefanoric: #qconlondon nodejs anti-patterns: replace a long list of arguments with an options object.

@stefanoric: #qconlondon nodejs anti patterns: use the revealing pattern instead of using the 'new' operator.

@timanderson: Modules can be too big, but no module is too small. Tip from Node.js session #qconlondon

@lukeb0nd: Use NPM scripts instead of gulp/grunt. Know if/when to switch though #nodejs @pgte @ igorsoarez @YLDio #qconlondon

@lukeb0nd: Use NPM scripts instead of gulp/grunt. Know if/when to switch though #nodejs @pgte @ igorsoarez @YLDio #qconlondon

Tammer explores how to draw the lines between services. dealing with performance issues, testing and debugging techniques, managing a polyglot landscape and the explosion of platforms, managing failure and graceful degradation.

"Boring is beautiful" – Tammer Saleh

Microservices Chaos Testing at Jet

by Rachel Reese

Twitter feedback on this session included:

@mpaluchowski: Microservices provide a more even distribution of complexity. #qconlondon @ rachelreese

@silverSpoon: The tests were too uniform, there was no variance @rachelreese At #qconlondon https://t.co/tQiUlE45kA

@ocklund: @rachelreese on chaos testing. We should plan for chaos, because world is chaotic. Design for failure then emerges. Test in prod #qconlondon

Test Driven Microservices
System Confidence through Journeys, Traces & Contracts...
@russmiles
QCon
QCon
Test Driven Microservices
System Confidence through Journeys, Traces & Contracts...
@russmiles

Test-Driven Microservices: System Confidence

by Russ Miles

Twitter feedback on this session included:

@mpaluchowski: Distributed monolith, worked reasonably badly in one place, now so in many places. #qconlondon @ russmiles

@oli_codes: Microservices .... "a monolith but distributed" HAHAHA @russmiles #qconlondon

@mpaluchowski: In our industry all we do is create our own problems. #qconlondon @russmiles

@mpaluchowski: The complexity of microservice systems should scare us. Focus on testing. #qconlondon @russmiles

@moogster31: mantra: "focus on testing the interface, not the implementation" #qconlondon

@mpaluchowski: The way to communicate and comprehend microservice systems are narratives. #qconlondon @russmiles

@kamkorz: #qconlondon In #microservices cognitive overhead matters, size does not. Well said.

The Microservices and DevOps Journey

by Aviran Mordo

Twitter feedback on this session included:

@robyoung26: "Don't add something to your architecture until you are feeling the pain of not having it" #qconlondon @aviranm

@mpaluchowski: Solve only problems you have. Don't just add technologies. #qconlondon @ aviranm

@mpaluchowski: Arrows in system schemas are failure points. #qconlondon @aviranm

@fabianpiau: The size of the microservice is the size of the team who is building it @aviranm #qconlondon

Business Mapping: Building an Agile Organization

by Chris Matts & Tony Grout

Twitter feedback on this session included:

@Joe0reilly: "We don't always need new things, we need to do the old things really well" Best phrase of the week from @tonygrout #qconlondon

@paulacwalter: Engagement and explaining why is key to prevent agile transformation being another management fad @tonygrout @PapaChrisMatts #qconlondon

@paulacwalter: Role play on agile decision making makes a good point @tonygrout @PapaChrisMatts scaling #agile #qconlondon

@KingPrawnBalls: Make transparent across your org the EPICS everyone is working towards.Agile in solos isn't an agile org. #qconlondon /Lloyds transformation

Culture Eats Principles for Breakfast

by Ian Dugmore & Jonathan Smart

Twitter feedback on this session included:

@techiewatt: The most dangerous phrase in the language is, 'We've always done it this way.' -Grace Hopper via #qconlondon talk by @iandugmore @ jonsmart

@danielbryantuk: Don't scale Agile first - de-scale the work first @jonsmart #qconlondon

@timanderson: Descale to improve velocity, eg reducing size of a dev team from 40 to 10 improved productivity at Barclays #qconlondon

@robyoung26: "Your practices are a function of your principles, given your context." @iandugmore @ jonsmart #qconlondon

@robyoung26: "primary measure of success for interactions between control functions (e.g. change control) is happiness" @iandugmore @jonsmart #qconlondon

@robyoung26: "You can't release anything until the cost of release is lower than the value you're trying to release" @iandugmore @jonsmart #qconlondon

@timanderson: Barclays gave some form of Agile training to 26,000 people last year #qconlondon

@robyoung26: "To effect cultural change, learning anxiety needs to be lower than survival anxiety" @ iandugmore @jonsmart #qconlondon

Papers We Love - London
Reading and Talking about Papers.
meetup.com/Papers-We-Love-London
QCon
QCon
QCon LONDON

Distributed Systems in Practice, in Theory

by Aysylu Greenberg

Alex Blewitt attended this session:

One useful nugget; if you have systems that may fail, building in a lease (pull request heartbeat) and then disconnecting a client whenever heartbeat failures occur is a way to main overall stability; especially if the client's job is re-submittable. And of course, a multi staged pipeline can be easily scaled if you separate the stages with queues and then have multiple consumers; but that's distributed scaling 101.

Not Quite so Broken TLS Using Unikernels

by Anil Madhavapeddy

Luke Bond attended this session:

I returned to the CS track to feel overwhelmed all over again, yet Anil managed to make a very difficult subject enjoyable and more-or-less understandable. Anil showed us how much of a mess the C code for very important libraries such as SSL can be, and how remarkable it is that they work as well as they do, despite the constant stream of exploits being announced. He proposes unikernels as a technology to enable us to write minimal library operating systems for our apps, rewriting these low-level libraries in type-safe, memory-safe and testable high-level languages to make the internet more secure.

Twitter feedback on this session included:

@silverSpoon: Who uses TLS/SSL? -all hands up- Who understands it? - two ppl At @avsm talk #qconlondon

@robertharrop: @avsm extolling the virtues of ASN.1 #qconlondon https://t.co/jL5LTsN0Hn

Rust: Systems Programming for Everyone

by Felix Klock

Luke Bond attended this session:

Returning to the CS track for more confusion, I was again pleased that this talk was very clear and understandable. Felix is an energetic and enthusiastic speaker and very engaging. Not being familiar with Rust some of the details were over my head but I was able to get the gist of most of it. Enough to tell me that I need to go and learn some Rust, it looks great.

Alex Blewitt attended this session:

Given that Rust uses extensively checked static compilation to verify that objects are not shared or mutated other than expected, and that the lifetimes of objects are tied to lifetimes in the code, this was a great overview (for me) to the subtleties in the way that the Rust applications work.

Twitter feedback on this session included:

@alblue: Comparing #RustLang 's crossbeam MPSC benchmark to Java and Scala's implementations @ pnkfelix at #QConLondon https://t.co/MYg9zdQf9S

@alblue: The parallelism in #RustLang article that @pnkfelix mentioned at #QConLondon is here: https://t.co/JUvqzsvcYK

@alblue: "If you remember nothing else from this talk, remember this: #RustLang cargo is really simple to use" @pnkfelix at #QConLondon

@alblue: "Semantic versioning is really important" — @pnkfelix on the importance of semver in #RustLang at #QConLondon https://t.co/KHRowAAdUi

Successful Go Program Design, 6 Years On

by Peter Bourgon

Twitter feedback on this session included:

@danielbryantuk: Advice on @golang repo structure from @peterbourgon at #qconlondon https://t.co/kRe0o7QmTE

@alblue: "If you remember no other slide, remember this one: make dependencies explicit" @ peterbourgon at #QConLondon https://t.co/YQ1JV2VSsA

@danielbryantuk: Key message I'm taking away from @peterbourgon's #qconlondon @golang talk is don't cargo cult stuff from other langs https://t.co/RAO0etwB7F

@danielbryantuk: Dependency management recommendations for @golang by @peterbourgon #qconlondon https://t.co/s9DH0qVpES

@danielbryantuk: Use 'FROM Scratch' if deploying @ golang via @docker @peterbourgon #qconlondon

The Case for Bringing Swift to the Server

by Chris Bailey & Patrick Bohrer

Alex Blewitt attended this session:

IBM have invested a lot into Swift, on the mobile side for the applications in which they are partnering with Apple, but also on the server side on Linux. Chris does a lot of work with open-source projects, such as Node.JS and Java already, and he and the team are working on porting libdispatch (also known as Grand Central Dispatch) to Linux, so that there can be portable multithreaded code running on OSX, iOS and Linux platforms. Patrick introduced IBM's Swift language sandbox on Linux, where each time an application is run the code is packaged up and deployed as a Docker container before compiling, running, and showing the results. He also quickly demonstrated Open Whisk which allows Lambda-style computation (writing in languages including Swift) to quickly deploy and upload snippets to the cloud.

Burnout

by John Willis

Luke Bond attended this session:

John Willis from Docker shared a moving story about the suicide of a friend who worked in DevOps, and went on to share some statistics and studies about burnout and some resources for those who need help.

Magnus Ljadas' attended this session:

John Willis talked about burnout, and it's a thing not only in society but in our business. Turns out software people tend to be more receptible than the general population, especially the high achievers. A slippery slope that may go unnoticed until too late. Regular self assessment to monitor indicators, just as you would monitor any system, and to just be there for your friends. Important stuff. Be alert.

The take-away is: talk, listen, ask if people are okay and, above all, care. Also check out the Maslach Burnout Inventory: take the test and see if you are at risk.

Twitter feedback on this session included:

@charleshumble: Burnout is the canary in the coal mine. @botchagalupe #qconlondon

@rachelreese: Some research shows burnout can be similar to PTSD. Self-test: https://t.co/NEiNpBamy8 #qconlondon

@ellispritchard: Great talk on recognizing burn-out by @botchagalupe at #qconlondon: I think if we look properly, we'll find it's an epidemic in our industry

Engineering You

by Martin Thompson

Jeroen Gordijn's attended this keynote:

He pled to go back to the basics of computer science. Even though learning a new JavaScript framework might be fun, the knowledge you gain will be obsolete in a matter of days, or weeks. However, learning about algebra, algorithms and mathematics will give you knowledge that will serve you for years. In IT we sometimes lose ourselves in technology, … instead of solving the problem the business has. A solution doesn't always need to be the best, most beautiful solution, but it needs to fulfill business requirement. We should not forget that our main focus should be our customer/business.

We should focus on architecture principles, like loose coupling and tight cohesion. …

He shows that there we in IT are quite slow on picking on the basics. In 1968 there was a conference where agile was already described with the following description:

The design process is an iterative one:
1. Flowchart until you think you understand the problem.
2. Write code until you realize that you don't.
3. Go back and re-do the flowchart.
4. Write some more code and iterate to what you feel is the correct solution.

Another great quote was made by A. J. Perlis in 1968, stating "A software system can best be designed if the testing is interlaced with the design instead of being used after the design", effectively describing TDD.

Twitter feedback on this session included:

@dosaki: Agile development process described back in '68 #qconlondon https://t.co/kQsnPB10o0

@andyhedges: Good engineers care about algorithms and data structures @mjpt777 at #qconlondon (AH: surprising how many people miss this)

@andyhedges: Some really strong points about abstraction from @mjpt777 at #qconlondon, the wrong abstracts REALLY hurt your software.

@timanderson: An ORM is the wrong abstraction for a database says Martin Thompson #qconlondon - think set theory as as OO

@andyhedges: The first rule of Abstration Club is don't abstract @mjpt777 #qconlondon

@laarchy: What should you learn as a Software Engineer ? The answer by @mjpt777 at #qconlondon https://t.co/OrDPMCdu77

@oli_codes: You have two ears and one mouth, use them in that proportion. Great quote about being a good software engineer. #qconlondon

@eyads: The book any developer could have written. #qconlondon @mjpt777 https://t.co/tyuqHpvhPy

Lead the Revolution by Being Ordinary

by Katherine Kirk

Twitter feedback on this session included:

@tastapod: Narcissism, Psychopathy, Machiavellianism: no the Dark Triad is not a Marvell comic but psychological phenomena, says @kkirk at #qconlondon

@tastapod: Loving @kkirk describing blame as an adrenalin-fuelled psychopath trip! #qconlondon

@jonsmart: Stress = Stupidity (cortisol) + Power (adrenaline). Not a great combination. Calm the * down. Change our reaction. @kkirk #qconlondon

@tastapod: .@kkirk just used the word "compassion" in a talk about people and coaching. I don't hear this nearly enough. #qconlondon

@KevlinHenney: I have to change the word 'compassion' to 'de-risking the people problem' when dealing with upper management.— @kkirk #QConLondon

@ocklund: @kkirk In new world of continuousness: "Gratitude is fearlessness. What's here that I can work with" #qconlondon https://t.co/1zPHdVjWjc

Making a Sandwich: Effective Feedback Techniques

by Dan North

Peter Liljenberg attended this session:

One of my most anticipated talks during the week was Dan North's "Making a sandwich". I hade very high expectations for this talk, and Dan managed to exceeded them (as usual). Dan talked about giving feedback, how feedback in itself is a system and why we should do it. Giving and receiving feedback (which is really just to say 'Thank you') is, in my opinion, one of the hardest skills to master and we should really practice a lot! Dan presented some useful techniques and tricks, but you should really watch this yourself!

merybere attended this session:

- Ask for feedback
- to improve
- for help
- for recognition, feel good
- We offer feedback
- to improve the system of work, as a team
- to model a culture, (when new people comes to a team it helps)
- to control others (let me give you some feedback...)
- to demonstrate our superior knowledge

Example: go and sit with people, observe how they do things, ask what they are doing is, start a conversation and being interested

Twitter feedback on this session included:

@nora_js: Dan North relating traditional workplace feedback to Systems Theory, putting it in terms engineers can understand :) @tastapod #qconlondon

@nora_js: You as a human are a system who relies on feedback. @tastapod #qconlondon

@mylenereiners: Who has an annual review? Who goes eleven months back to improve? @tastapod #qconlondon

@aviranm: Timing is everything. Small and frequent feedback is better than large and infrequent. @tastapod #qconlondon https://t.co/GghdKoW5Wj

@charleshumble: Go and sit with the people who use your product. Each time they sigh write down what they were doing. @tastapod #qconlondon

@nora_js: And remember, when you receive feedback, always say: 'Thank You'. @tastapod #qconlondon

@ocklund: @tastapod Low-trust environments: Offer specific positive feedback, everything else will self-correct #qconlondon https://t.co/SCCaLYOz0a

Bitcoin Security: 1/10th Cent to a Billion Dollars

by Olaf Carlson-Wee

Twitter feedback on this session included:

@paulacwalter: Olaf Carlson-Wee: if your PC is slow be aware you may be mining Bitcoin for someone in another country #qconlondon

@techiewatt: Olaf Carlson-Wee #qconlondon there are generally 2 threats to your Bitcoins - attackers and you!

Building a Modern Security Engineering Team

by Zane Lackey

Twitter feedback on this session included:

@techiewatt: #qconlondon @zanelackey - surface security info for everyone, not just security teams!

@KingPrawnBalls: Put the security dashboards right where engineering teams are so they'll see it! Don't hide them away #qconlondon / @zanelackey

@KingPrawnBalls: Reward engineers' communication with the security team: t-shirts, gift cards, etc. Bootstrap the interaction! #qconlondon / @zanelackey

@techiewatt: #qconlondon @zanelackey contrary to populate belief "deploying quicker makes you more secure"

@KingPrawnBalls: Security in DevOps culture: 3 keys (1/3): 1) make security monitoring &amp; metrics available to all #qconlondon / @zanelackey

@KingPrawnBalls: Security in DevOps culture: 3 keys (2/3): 2) incentivize dialog with the security team (gifts, t-shirts, etc) #qconlondon / @zanelackey

@KingPrawnBalls: Security in DevOps culture: 3 keys (3/3): 3) your policies should not take away capabilities #qconlondon / @zanelackey

@MrAndyButcher: Security teams, don't be a blocker - you'll get worked around. Positive and insightful advice from @zanelackey just now #qconlondon

Microservices for a Streaming World

by Ben Stopford

Philip Carrington attended this session:

This talk looked at an add-on for Apache Kafka called KStreams that allow you to persist the latest version of a key so that it could be use by a microservice in combination with a stream to create other services. We also need to embrace decentralization.

KStreams can be used to make KTables that can be joined with data from a stream to enable querying for a microservice

Kafka has compacted tables that allow you to store the latest value for a key if you so wish!

Real-time Stream Computing & Analytics @Uber

by Sudhir Tonse

Twitter feedback on this session included:

@danielbryantuk: Interesting to see that @UberEng store steam processing data in ElasticSearch, rather than a typical DB. @stonse at **#qconlondon**

@OpenCredo: Which stream processing tools should you use? That depends on your use case @stonse at #qconlondon #NoGoldenHammer https://t.co/Aj4oIoMauq

@danielbryantuk: Listening to @stonse at #qconlondon, and I'm hearing that many stream processing tools are operationally complex (listen up tool makers :-))

Stream Processing with Apache Flink

by Robert Metzger

Philip Carrington attended this session:

New product in a way that will get its full release tomorrow (08/03/2016). It is promising to completely subsume batch by allowing windowing over "large" timescales by utilizing in memory and disk persisted aggregations as well as a host of other interesting features that other systems do not offer.

Google Dataflow is being made into an Apache incubator project called Apache Beam

Twitter feedback on this session included:

@OpenCredo: Using log streaming (via Kafka) allows easy horizontal scalability, as individual workers can pick events and process them #qconlondon

@PaulAnkho: Flink state management seems to resolve issues suffered using storm at production. How will it perform? Pretty cool! Need PoC #qconlondon

@mpaluchowski: You need to use #Kafka because there's no good alternative. #qconlondon @ rmetzger_

Using Technology as a Blind Long Distance Runner

by Simon Wheatcroft

Luke Bond attended this session:

Simon taught himself to run, blind, using the RunKeeper running app for his phone. Starting on a football field, using the audio feedback, he moved to closed suburban roads and then open motorways. Astonishingly, he uses only the combination of route memorization through under-foot feel, the pain of running into things, and the audio feedback of RunKeeper. He has since run marathons and is about to participate in a desert race using custom technology he has produced with the help of IBM.

Magnus Ljadas' attended this session:

What a heartwarming moment to watch Simon Wheatcroft, the blind ultra marathon runner, on stage with his guide dog slumbering at his feet. Being a mere marathoner myself I could not constrain my amazement over this man's achievements, let alone he's blind on top of that. Technically speaking, the solution that will enable him to run 126 km unassisted through the Namibian desert, is rather straight forward—a gps based beeper thingie that simply beeps when he strays off track. No drones, no real time room mapping, nothing fancy at all. I will start hallucinate anyway so I want it to be simple, he explained. Stupid simple solutions are the best.

Opinions about QCon

Tim Anderson's attended the conference:

This is a software development conference focused on large-scale projects and with a tradition oriented towards Agile methodology. It is always one of the best events I get to attend, partly because it is vendor-neutral (it is organized by InfoQ), and partly because of the way it is structured. The schedule is divided into tracks, such as "Back to Java" or "Architecting for failure", each of which has a track leader, and the track leader gets to choose who speaks on their track. This means you get a more diverse range of speakers than is typical; you also tend to hear from practitioners or academics rather than product managers or evangelists.

Impressions expressed on Twitter included:

@rvanbruggen: Hey @qconlondon - congrats on your 10y anniversary conference this week. Here's a #neo4j #graphdb as a present :) https://t.co/D2yUPVQgjR

@dthume: 5 minutes into #qconlondon and I'm as impressed as ever by how helpful and friendly the staff are.

@timanderson: Of all the events I attend, #qconlondon has the best food*. Way to a developer's heart? *possible exception for #monkigras

@russmiles: Glad to see the simple card voting system in effect! #qconlondon https://t.co/2OqjQ9Mo3f

@igatanasov: #qconlondon ten years later but better than ever

@fintanr: The breadth of talks at #qconlondon is really impressive, lots of videos to watch post conference

@danielbryantuk: Every time at #qconlondon I think I have my schedule set, but the track hosts convince me otherwise :-) // cc @wesreisz

@ellispritchard: Last thing I expected to see at #qconlondon - amazing veg, including oxalis tuberosa &amp; Stachys affinis! https://t.co/d6BDoQaFhq

@danielbryantuk: However you do it, share your knowledge. I'm convinced this is how the industry progresses @charleshumble #qconlondon Amen to that!

Takeaways

Luke Bond's takeaways were:

QCon strives to appeal to a broad range of technologies and levels of developer expertise. Most of the talks reaffirmed some of my previous knowledge, but I was challenged and intrigued by the more advanced computer science talks. Overall I found it really valuable to get the "lay of the land", to see what technologies people are interested in today and using in production. It is also a good opportunity to meet like-minded people and "network".

Jeroen Gordijn's takeaways were:

My take away of this conference is that I should read more articles with a focus on software science, instead of all the blogs concerning a specific technique.

Peter Liljenberg's takeaways were:

Most of the interesting sessions I attended during the week were about failure, and how to handle failures. Quotes like "Failure is inevitable", "Failure is an opportunity to learn" and the importance of building an architecture that can manage failures were common topics. Migrating from a monolithic application to a more micro service oriented architecture were also popular.

Takeaways from QCon London included:

@paulmfarrar: Sum up #qconlondon in one word? Collaborate!

@urbanisierung: Thanks @qconlondon for a great conference again! For me it was so far the most inspiring of all I have participated! #qconlondon

@kriz_davison: Thankyou #qconlondon for yet another amazing conference. As always its got me energised and broadened my horizons. But now my brain hurts!!

@herder: So, home from #qconlondon.Time to look at the videos of all the great stuff I didn't see live!

@nginx: Thanks for a great event #qconlondon. It was awesome to meet so many people interested in modern web architecture and #microservices.

@kennethmac2000: Going to a conference for 5 days in a row was hard work, mentally tough, but rewarding! #qconlondon

Conclusion

QCon's focus on practitioner-driven content is reflected in the fact that the program committee that selects the talks and speakers is itself comprised of technical practitioners from the software development community.

QCon London is one of a global series of events held around the world. Hot on the heels of QCon London we held QCon São Paulo on March 28th-April 1st, with QCon Beijing just around the corner starting on April 21st. The next English language QCon is New York starting on June 13th, followed by San Francisco on November 7th.

QCon London will return on 6th-10th March 2017.: