RabbitMQ
Open Source Enterprise Messaging

October 2009

RabbitMQ – Messaging that just works

Executive Summary

RabbitMQ is the market leader in AMQP with several hundred production deployments and hundreds more in development. We have the largest open source community of AMQP practitioners in Java, JMS, Ruby, C, C#, .NET WCF, Python, and more, and are renowned for being user friendly. We are unique in being multiprotocol, with adaptors for XMPP, SMTP and HTTP for lightweight web messaging.

Recently Canonical made the decision[1] to distribute RabbitMQ in Ubuntu 9.04, alongside Eucalyptus, for cloud messaging. This, along with other distributions like Debian, gives us leading penetration in the Linux market. We are also the most deployed AMQP broker on Windows, with RabbitMQ.NET edition, and on the cloud, notably Amazon and Engineyard.

Introduction to AMQP and RabbitMQ

AMQP is the open standard protocol for business messaging that has widespread industry backing from companies ranging from JPMorgan and Credit Suisse, to Cisco and Microsoft. RabbitMQ provides a complete, secure and highly reliable AMQP Enterprise Messaging system. The RabbitMQ client libraries and broker daemon can be used together to create an AMQP network, or used individually to bring the benefits of RabbitMQ to established networks. The broker interoperates with other brokers on the 0-9-1 AMQP protocol.

The product is on version 1.7.0 and has been available since 1 Feb 2007 under the open source Mozilla Public License. A large number of customers have adopted the software in production, especially on the Amazon EC2 cloud, and engaged the RabbitMQ team for commercial services including support, training and custom development.

Well Supported Across All Platforms

RabbitMQ is fast becoming the messaging server of choice in multiple sectors because it offers a consistent and interoperable approach to messaging across any platform.

RabbitMQ is equally suited to server side, browser, mobile phone or cloud cases. It is seen by many as the leading open messaging broker due to its quality, support, thriving community, number of contributors, and breadth of use cases - see http://www.rabbitmq.com/how.html for more info, supported platforms, clients and protocols *e.g.* HTTP, XMPP, STOMP, SMTP.

Background on AMQP

AMQP stands for Advanced Message Queuing Protocol. It is a completely open protocol that was released in 2006 and since then has seen serious adoption across all verticals, from financial services to large scale web application providers. AMQP is a complete solution for all categories of messaging ranging from pubsub and content-based routing, to reliable and transactional transfer of large data sets. Unlike JMS, AMQP is a binary wire protocol enabling testable interoperability across any language and platform. This translates directly into much lower integration costs since, like TCP or HTTP, messaging becomes a simple matter of wiring. AMQP is a modern design, much more like a switching exchange than like a database, and not held back by technology decisions made in the 1990s.

Adoption has rapidly increased in the last year as the 'working group' of supporting vendors and end users reached critical mass - see http://www.amqp.org for details. RabbitMQ has been a member of the group since early 2007, and the chairman of the working group, John O'Hara at JPMorgan, would be happy to attest to the strength of our contributions to the specification and its adoptions.

For reading on AMQP and RabbitMQ, please see http://www.rabbitmq.com/how.html, especially the presentation at Google UK, and Open Enterprise's series.

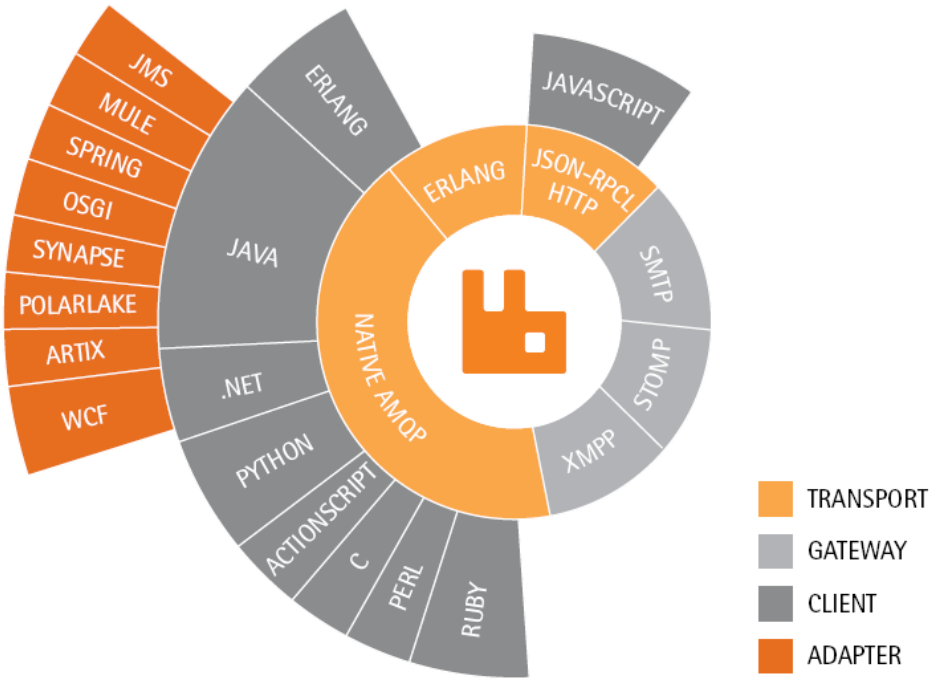
RabbitMQ supports AMQP 0-8 and 0-9-1. For reading on the AMQP roadmap see: http://www.rabbitmq.com/specification.html and http://en.wikipedia.org/wiki/AMQP.

[1] Link http://www.emediawire.com/releases/2009/4/prweb2340344.htm

Packages Supported Commercially

Supported packages are QAd to commercial grade for every RabbitMQ release by the engineering team. The main platforms are here: http://www.rabbitmq.com/download.html. They include mainstream operating system packages such as Windows, Ubuntu, Fedora, Debian and SUSE. In addition, RabbitMQ community builds exist for MacOSX, BSD, Gentoo and OpenSolaris. We are working to integrate these into official distributions. We are willing to support customised non-standard builds on a commercial basis.

The following diagram illustrates the RabbitMQ universe of clients. Note that "Java" includes POJO, JMS, and OSGi, and that .NET includes C# and WCF.



Broker Performance

RabbitMQ has been certified by Intel, our financial services testing and performance partner. Please contact us for details – all performance is highly use case dependent especially in the presence of slow clients. See also this news group posting by us that contains an analysis of the performance results: http://www.nabble.com/rabbitmq-performance-td14103858.html.

The system is able to scale well up to substantial loads – full details are in the conformance section of the RFP response. Roundtrip latency of sub 500µs is reasonable on a 1GigE LAN, when RabbitMQ is not overloaded, and loads compare well with other leading products known for high performance. RabbitMQ is not a 'latency buster' in the sub 50µs class.

Technology and Architecture

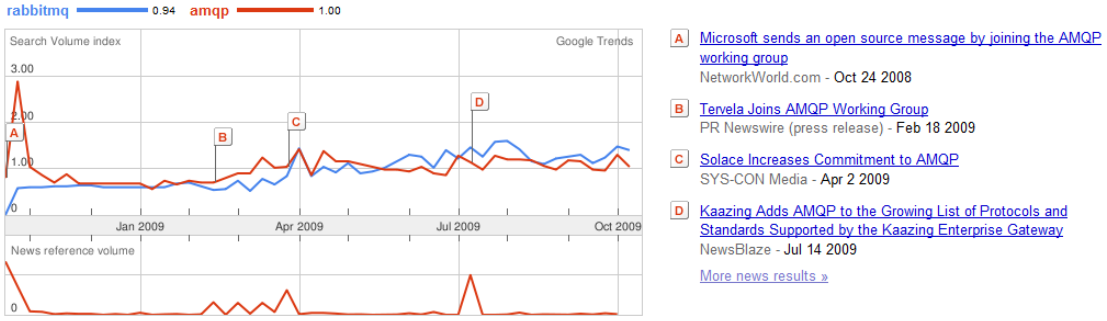
The core RabbitMQ broker is extraordinarily compact at only 12,000 lines of code, which as one of our customers put it "makes it very hard to hide bugs" and more importantly lowers the cost of management and maintenance noticeably. This is because the core engine is written in Erlang/OTP, an 'always available' platform that has been in production use in carrier grade telcos since the late 1990s. RabbitMQ inherits many attractive features from this including self-healing cluster management and extensive tooling. Because RabbitMQ implements the open AMQP protocol there is no need to know Erlang and many users prefer to manage the broker using Java or Ruby, e.g. http://brainspl.at/articles/2009/04/30/you-got-ur-ruby-in-my-erlang.

For further information on the architecture of RabbitMQ we recommend the second presentation at Google London linked to from: http://www.rabbitmq.com/how.html.

Leading Open Source Adoption

By the nature of it being an open source project we do not have complete data on the volume of users or the nature of their use cases.

Over the last six months interest has grown steadily, and RabbitMQ is now the most searched for AMQP implementation on Google.



Source: Google Trends

We have seen RabbitMQ become the market leading open source messaging system over the course of the past year. Compared to other AMQP implementations we can confidently say that we have the most active community with about 500 members currently active. RabbitMQ has been downloaded by an estimated 15,000 people and this is set to rise rapidly now we are packaged with Ubuntu, Debian, Orbited, Nanite and Lift.

RabbitMQ Commercial Services

The RabbitMQ team has provided commercial consulting and support services since 2007. Many companies and teams use RabbitMQ in production – we estimate the number to be at least fifty known good cases who have contacted us about current use, plus many unknown due to the nature of the open source distribution.

RabbitMQ Use Cases

Many people talk about their use of RabbitMQ on the web or report it on our mailing list. The web page here gathers links to these sources: http://www.rabbitmq.com/how.html. Below we list some examples of how people are using RabbitMQ.

Publish / Subscribe Messaging

This is a very common pattern and typically uses RabbitMQ's AMQP direct exchange. It is used for financial services data integration. For example, several banks and brokers are sharing real time data between Excel spreadsheets using RabbitMQ.NET. These systems are high scale and high throughput relative to mainstream use. The same solution typically connects to other systems, either for grid computing, to back end systems for integration with Java, Python and so on.

The pubsub pattern is also used by numerous 'Web 2.0' companies. Instead of dealing with market data, they add event management, group chat, and "twitter like" capability to their applications (including several 'twitter clones').

Reliable Delivery

RabbitMQ's ability to persist data and recover from failures is used for applications such as eventual delivery, event logging and audit. For example several companies use the SMTP adaptor for email filtering and processing.

The same capability can be combined with transactions to provide acknowledgements of delivery. RabbitMQ has also been used with ESBs, mainly for Java service architectures and reliable integration.

Disaster Recovery and Replication

At least one company we know of is using RabbitMQ to keep its several data centres

reconciled across WAN links. Essentially the model is a standard bulk write behind pattern, of interest mainly due to its global reach. Several companies are also investigating global replication of trading desk data in close to real time.

An interesting new example – not yet in production – is from a large software vendor in the financial sector that is aiming to deploy RabbitMQ in a counterparty payments service to replace their use of e.g. SWIFT, which is expensive and inflexible. This would be the first true and large scale B2B use case for AMQP in the banking sector: each bank would use the service to make payments.

Scaling Applications, Work Offloading and Batch Processing

Moving away from the vanilla 'standalone messaging' use cases, we see a lot of users who integrate RabbitMQ into their systems for grid computing style workload distribution.

The need to scale turns out to be common in the Python and Ruby communities. Around seven different toolkits and frameworks have been implemented to provide batch processing of web requests and other work, all using RabbitMQ. Some of these have been integrated with application stacks such as Django.

One popular example is Nanite, which has around 650 followers on the Github repository (link http://github.com/ezmobius/nanite/tree/master. Nanite is emerging as a mainstream way to scale Ruby applications using RabbitMQ especially on the Engineyard hosting service and Amazon cloud. Others include Workling, Peloton, as well as custom cases such as this one: http://en.oreilly.com/rails2009/public/schedule/detail/8519.

Real Time Data in the Browser

With the rise of AJAX style web applications that use JavaScript or ActionScript to enrich the browser experience, the need has grown to extend the reach of messaging across the open web to provide 'push' updates over HTTP. RabbitMQ supports multiple patterns here. On the one hand it is a tool of choice for people using speciality real time HTTP push tools such as Orbited, Kaazing and Bayeux. Of these Orbited is probably the most used with RabbitMQ.

Many users prefer to use RabbitMQ's built-in HTTP client which enables JavaScript to talk directly to a RabbitMQ in a JSON/RPC style. We also support other scripting clients eg Flex.

The advantage of using RabbitMQ for these cases is primarily ease of management – it is attractive to use a single messaging infrastructure for all needs from client side to back end.

Cloud Computing: Management, Provisioning, Scaling and On-Demand Processing

RabbitMQ is designed to scale transparently, using the underlying autonomic capability of Erlang/OTP. It has taken hold in the 'cloud computing' sector in a number of areas. We have several large users who provide cloud services that themselves host many thousands of users on Amazon EC2 and other infrastructure providers. They all use RabbitMQ for provisioning, monitoring and management of live environments. Elements are here: http://brainspl.at/articles/2009/04/30/you-got-ur-ruby-in-my-erlang.

A second category of cloud users run RabbitMQ on EC2 to scale their applications, or as a "cloud burst" or "cloud gateway" mechanism to communicate between Amazon services, such as EC2 and S3, and their on-premise systems. For example, one company using RabbitMQ on EC2 process terabytes of data per day for optimization of advertising processing algorithms; another for 'click counting' and page ranking.

Finally, with the bundling of RabbitMQ on Ubuntu 9.04, alongside the Eucalyptus cloud software, companies are starting to create 'private clouds' using open source software. This is very exciting but quite new, with real use anticipated to grow after the next release (9.10), see http://news.cnet.com/8301-19413_3-10168951-240.html.

Many of these cases are shown here: http://delicious.com/alexisrichardson/cloud+rabbitmq.

The NASA "Nebula" Cloud uses RabbitMQ: http://nebula.nasa.gov/services.

An example approach to management and monitoring is described here: http://enthusiasm.cozy.org/archives/2009/02/listening-to-the-system.

Security and Multisite VPN Solutions

RabbitMQ is embedded inside the CohesiveFT product "VPN-Cubed" which enables a federation of VPNs to be set up quickly, providing multi-site redundancy and a secure bridge to the cloud: http://www.cohesiveft.com/vpncubed/. The use of RabbitMQ is as a secure pubsub system that works even when multicast is forbidden e.g. WAN and Cloud.

The Real Time Web and Content Streaming

The BBC is developing a service called Feeds Hub that extends RabbitMQ with additional content management capability such as integration with Atom and RSS. This is described here: http://www.bbc.co.uk/blogs/radiolabs/2009/04/introducing_bbc_feeds_hub.shtml and http://www.lshift.net/blog/2009/05/08/untangling-the-bbcs-data-feeds. Media streaming will be a natural extension of this. Another one of our customers is in the set top box business.

Other Protocols: XMPP, STOMP, File Transfer

RabbitMQ supports an XMPP gateway which we used to build a prototype IM pubsub system called Rabbiter (a toy twitter). This can also be used for monitoring with an IM interface. The RabbitMQ STOMP gateway is used for lightweight integration and in ActiveMQ replacement scenarios. RabbitMQ has been used for large file streaming too. For example, Nanite supports chunking of machine images for delivery.

Mobile Phones and GPS

Customers are exploring applications that use RabbitMQ on the iPhone, Android, and Windows Mobile platforms. We also have a customer looking at using GPS and GSM with RabbitMQ to manage a fleet of 30,000 cars roving around North America.

International Data Delivery Network

We are working with a customer to develop a global data delivery network (DDN) for climate change research. The customer represents 55 separate institutions in the US that form part of a larger network of 1,000 institutions around the world.

Each institution conducts its own experiments and generates data in the form of photographs, movies, documents and databases. Terabytes of data are generated daily and shared across the internet and private government-funded data networks.

RabbitMQ forms the backbone of the DDN. Institutions that are interested in a particular experiment or activity subscribe to it on the network. As data is generated and published it is sent as a stream of updates to software agents that notify the interested parties that, for example, 'this file has changed' or 'a new experiment has begun'. In effect, this is a social network for scientists – a 'twitter for data'.