Software Architecture for Big Data Systems

Ian Gorton
Senior Member of the Technical Staff - Architecture Practices

Ian Gorton is investigating issues related to software architecture at scale. This includes designing large scale data management and analytics systems, and understanding the inherent connections and tensions between software, data and deployment architectures in cloud-based systems.

I've written a book in 2006, Essential Software Architecture, published by Springer-Verlag. It sold well and has had several excellent reviews in Dr Dobbs and ACM's QUEUE Magazine. A 2nd Edition was published in 2011. I also co-edited 'Data Intensive Systems' which was published by Cambridge University Press in 2012. I've also published 34 refereed journal and 100 refereed international conference and workshop papers, with an h-index of 28.

Copyright

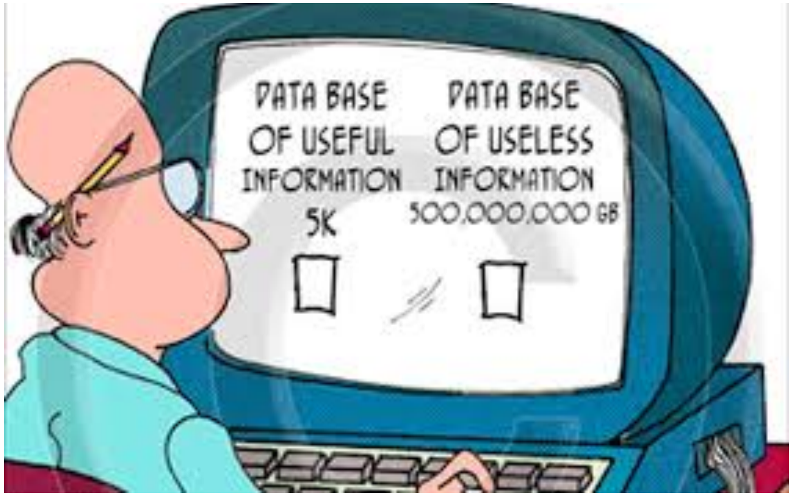
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0001080

Scale changes everything

WHAT IS BIG DATA?

FROM A SOFTWARE ARCHITECTURE PERSPECTIVE …



Some Big Data …

Google:

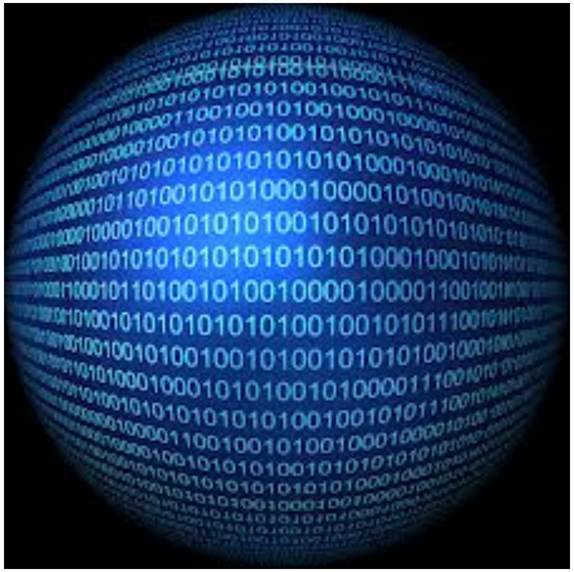
- Gmail alone is in the exabyte range

Salesforce.com

- Handles 1.3 billion transactions per day

Pinterest.com

- 0 to 10s of billions of page views a month in two years,
- from 2 founders and one engineer to over 40 engineers,
- from one MySQL server to 180 Web Engines, 240 API Engines, 88 MySQL DBs + 1 slave each, 110 Redis Instances, and 200 Memcache Instances.



http://highscalability.com/blog/2014/2/3/how-google-backs-up-the-internet-along-with-exabytes-of-othe.html

http://highscalability.com/blog/2013/9/23/salesforce-architecture-how-they-handle-13-billion-transacti.html

http://highscalability.com/blog/2013/4/15/scaling-pinterest-from-0-to-10s-of-billions-of-page-views-a.html

Not so successful ….

Some first-wave big data projects 'written down' says Deloitte

Not enough data a problem for some, while Hadoop integration has proved tricky

By Simon Sharwood, 19 Feb 2014 Follow 3,278 followers

Transforming your business with flash storage

Consultancy outfit Deloitte reckons early big data projects have had to be written down because they failed, thanks in part to a "buy it and the benefits will come" mentality.

The source of failure was sometimes difficulty making open source software work and/or integrate with other systems, Deloitte Australia's technology consulting partner Tim Nugent told *The Reg*. Such failures weren't because the software was of poor quality. Instead, organisations weren't able to make it do meaningful work because they lacked the skills to do so. Integrating big data tools with other systems also proved difficult.

The attempt to develop those skills while also staying abreast of the many changes in the field of big data proved hard for some, Nugent said. Happily, vendors and services providers have since come up to speed and are making

Why Most Big Data Projects Fail + How to Make Yours Succeed

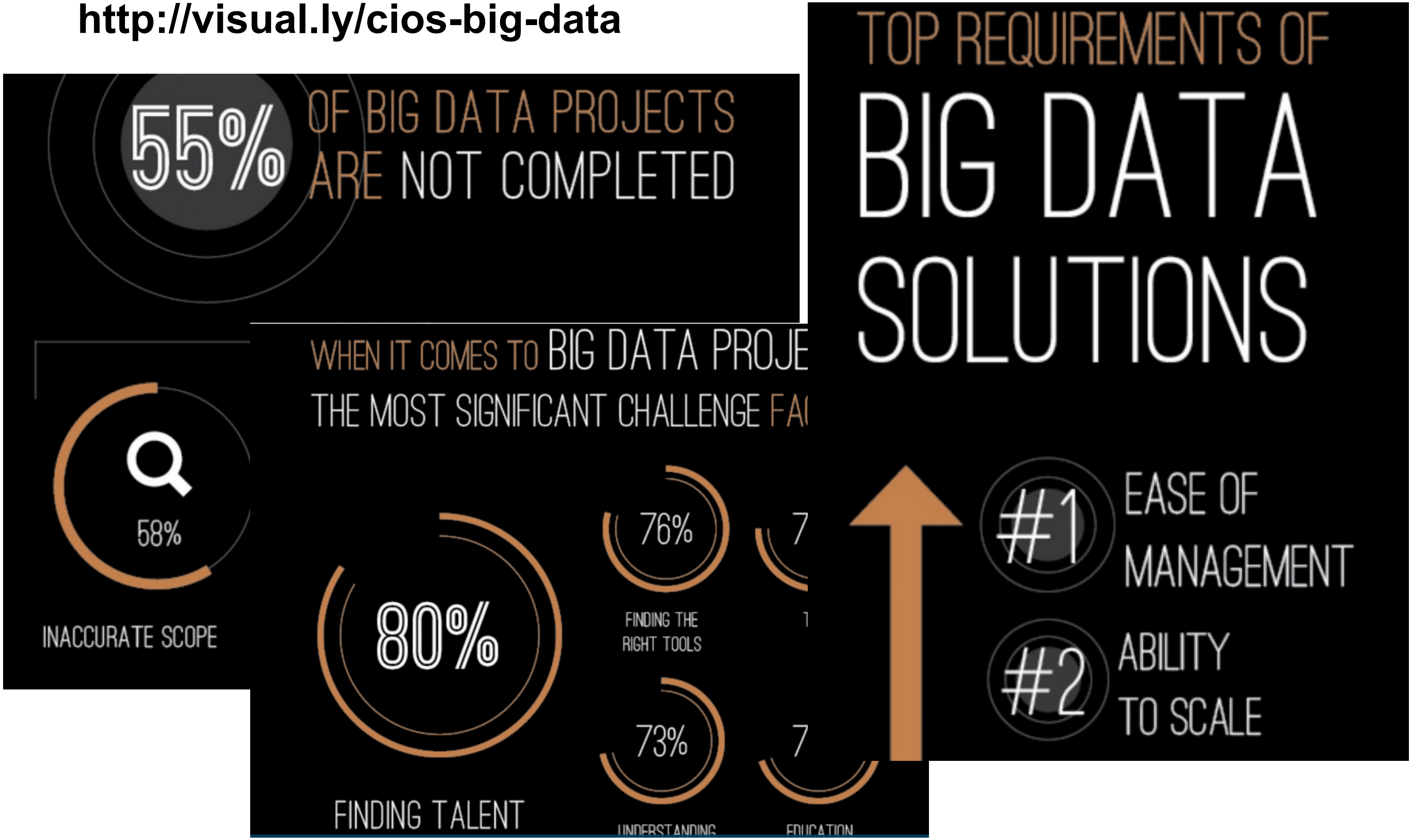
By Darin Bartik | May 14, 2013 Follow 185 followers

CXM Webinar: Deliver contextually relevant experiences across any channel, device or language

Big data is on the minds of just about everyone, with IT departments large and small grappling with exponentially growing volumes of both structured and unstructured data. But despite big data's place as a mainstream IT phenomenon, the bulk of big data projects still fail, as organizations struggle to find ways to capture, manage, make sense of and ultimately, derive value from their data and information.

- Lack of knowledge. Many of the technologies, approaches and disciplines around big data are new, so people lack the knowledge about how to actually work with the data and accomplish a business result.

Big Data Survey
http://visual.ly/cios-big-data



Big Data – State of the practice "The problem is not solved"

Building scalable, assured big data systems is hard

- Healthcare.gov
- Netflix – Christmas Eve 2012 outage
- Amazon – 19 Aug 2013 – 45 minutes of downtime = $5M lost revenue
- Google – 16 Aug 2013 - homepage offline for 5 minutes
- NASDAQ – June 2012 – Facebook IPO

Building scalable, assured big data systems is expensive

- Google, Amazon, Facebook, *et al.*
  - More than a decade of investment
  - Billions of $$$
- Many application-specific solutions that exploit problem-specific properties
  - No such thing as a general-purpose scalable system
- Cloud computing lowers cost barrier to entry ***– now possible to fail cheaper and faster***

NoSQL – Horizontally-scalable database technology

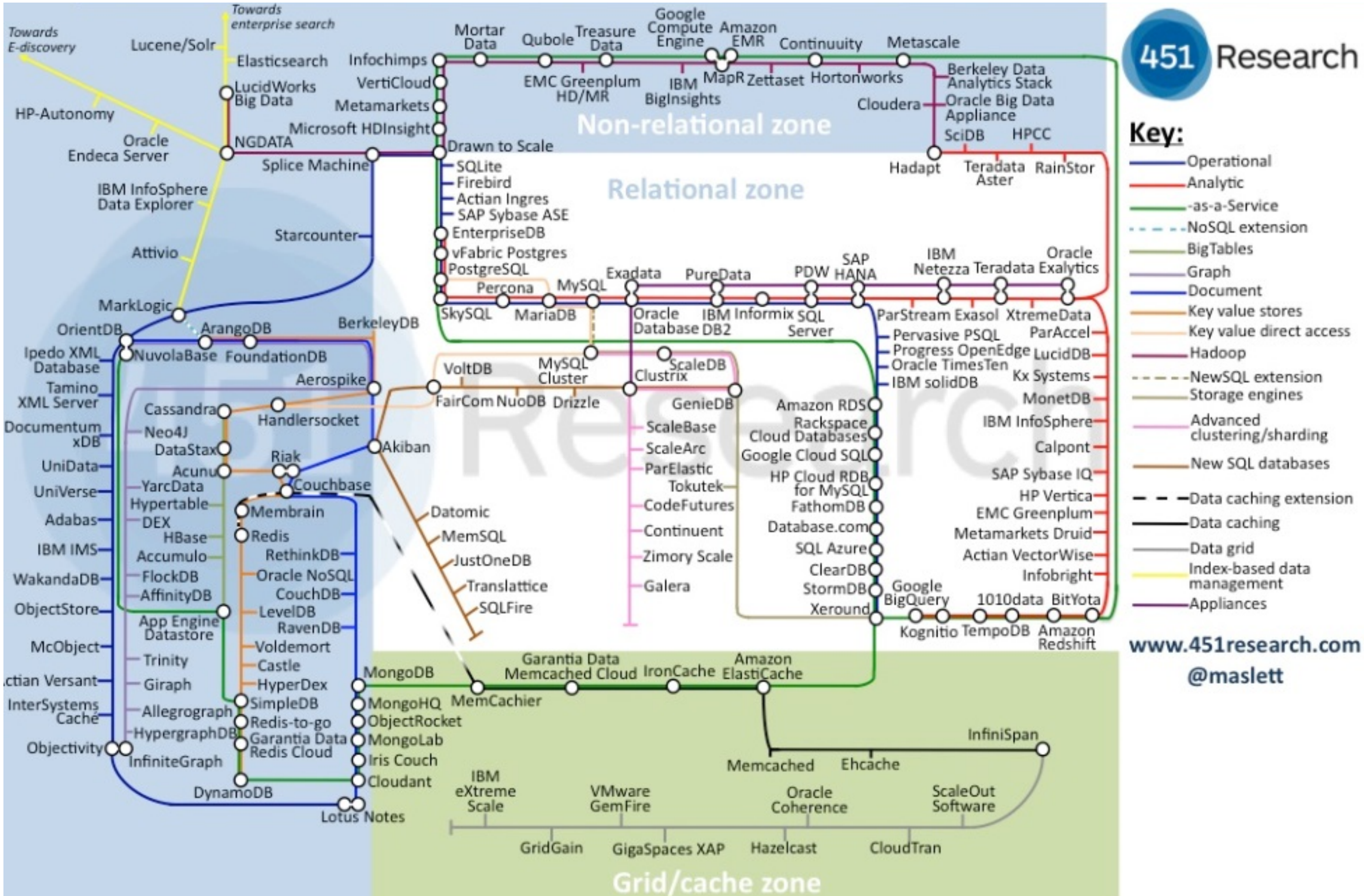
Designed to scale horizontally and provide high performance for a particular type of problem

- Most originated to solve a particular syster problem/use case
- Later were generalized (somewhat) and many are available as open-source packages

Large variety of:

- Data models
- Query languages
- Scalability mechanisms
- Consistency models, e.g.
  - Strong
  - Eventual

NoSQL Landscape



https://blogs.the451group.com/information_management/files/2013/02/db_Map_2_13.jpg

Horizontal Scaling Distributes Data
(and adds complexity)

Distributed systems theory is hard but well-established

- Lamport's "Time, clocks and ordering of events" (1978), "Byzantine generals" (1982), and "Part-time parliament" (1990)
- Gray's "Notes on database operating systems" (1978)
- Lynch's "Distributed algorithms" (1996, 906 pages)

Implementing the theory is hard, but possible

- Google's "Paxos made live" (2007)

Introduces fundamental tradeoff among "CAP" qualities

- Consistency, Availability, Partition tolerance (see Brewer)
- "When Partition occurs, tradeoff Availability against Consistenc Else tradeoff Latency against Consistency" (PACELC, see Abadi)

"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable"

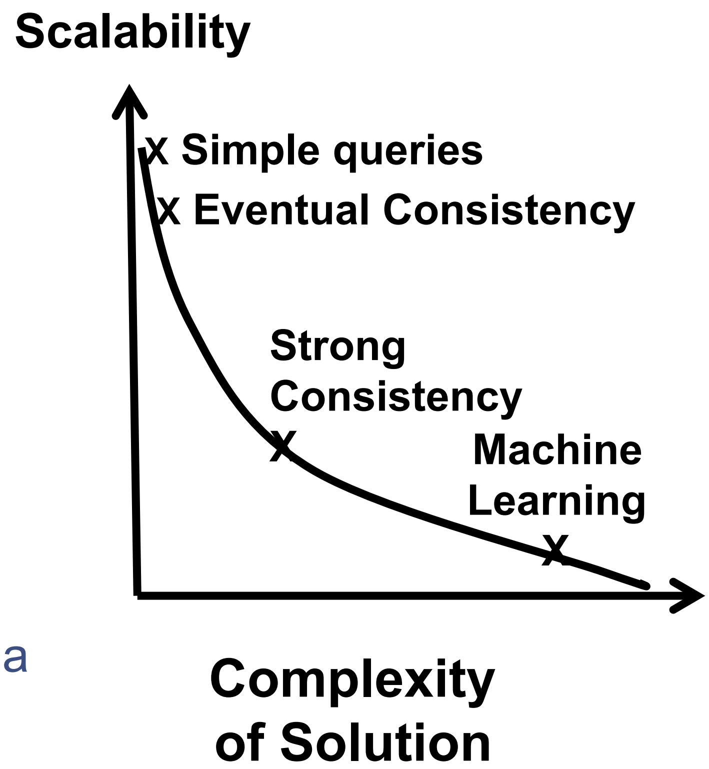
Rule of Thumb: Scalability reduces as implementation complexity grows

Workload

- # of concurrent sessions and operations
- Operation mix (create, read, update, delete)
- Generally, each system use case represents a distinct and varying workload

Data Sets

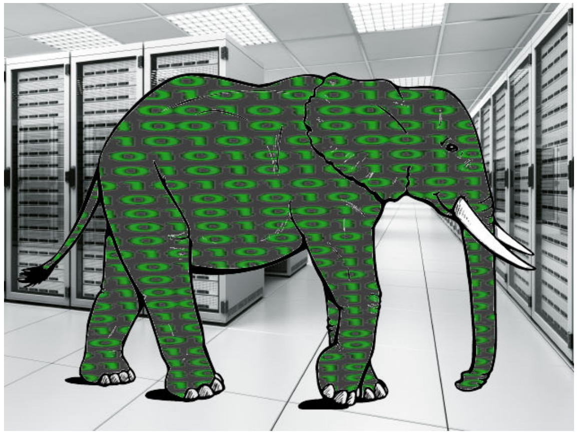
- Number of records
- Record size
- Record structure (e.g., sparse records)
- Homogeneity/heterogeneity of structure/schema
- Consistency



Big Data –
A complex software engineering problem

Big data technologies implement data models and mechanisms that:

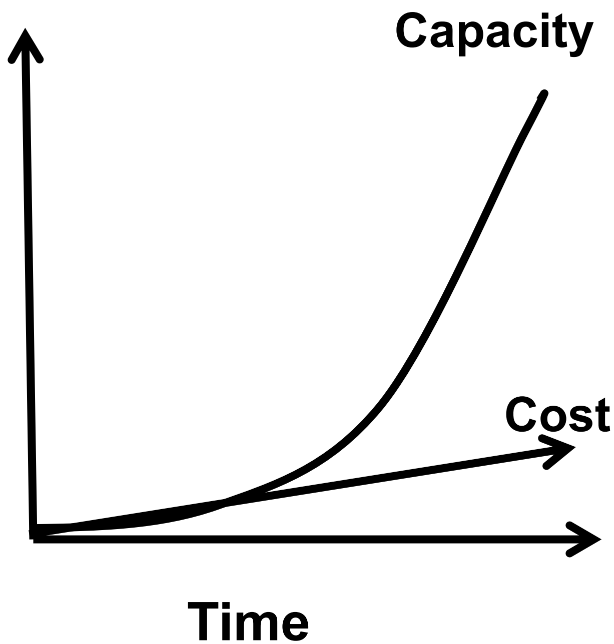
- Can deliver high performance, availability and scalability
- Don't deliver a free lunch
  - Consistency
  - Distribution
  - Performance
  - Scalability
  - Availability
  - System management
- Major differences between big data models/ technologies introduce complexity



Software Engineering at Scale

Key Concept:

- system capacity must scale faster than cost/effort
  - *Adopt approaches so that capacity scales faster than the effort needed to support that capacity.*
  - *Scalable systems at predictable costs*



Approaches:

- Scalable software architectures
- Scalable software technologies
- Scalable execution platforms

SO WHAT ARE WE DOING AT THE SEI?

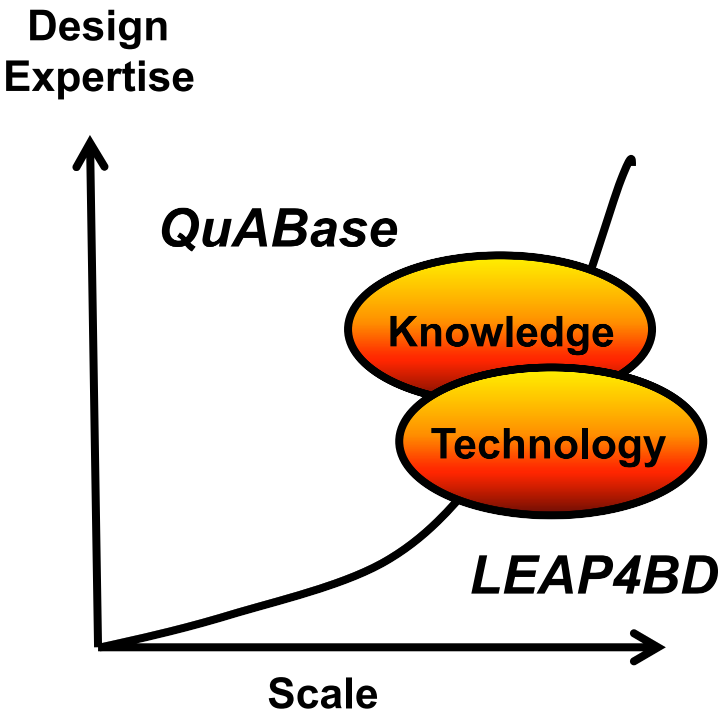
Enhancing Design Knowledge for Big Data Systems

Design knowledge repository for big data systems

- Navigate
- Search
- Extend
- Capture Trade-offs

Technology selection method for big data systems

- Comparison
- Evaluation Criteria
- Benchmarking



LEAP4BD

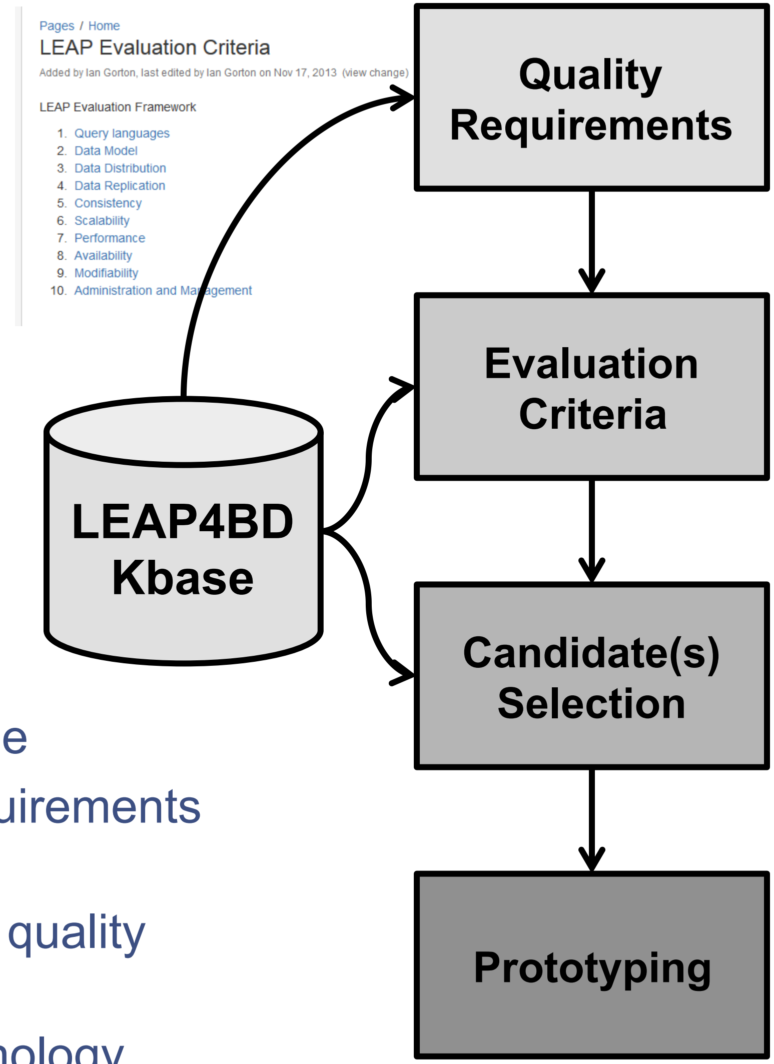
Lightweight Evaluation and Architecture Prototyping for Big Data (LEAP4BD)

Aims

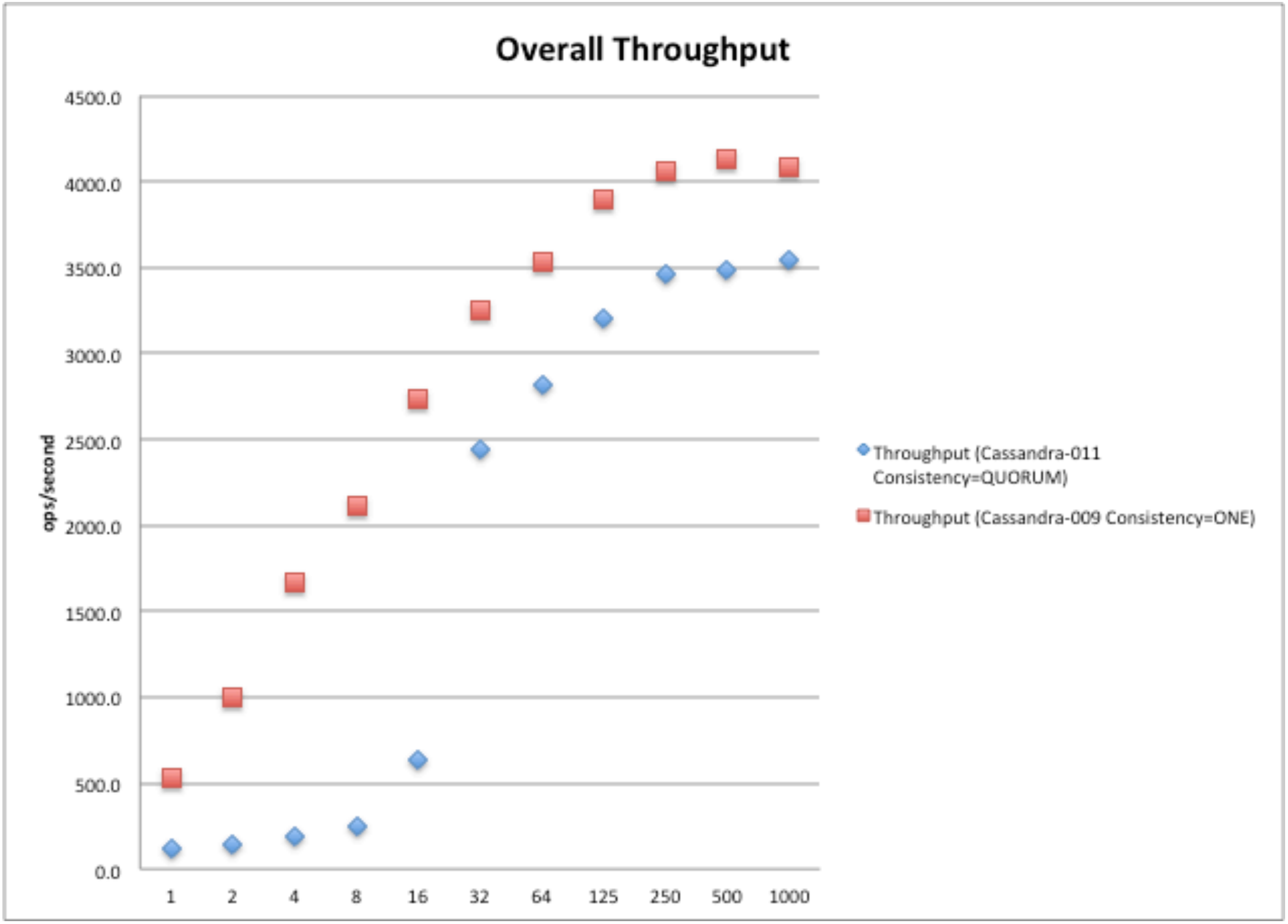
- Risk reduction
- Rapid, streamlined selection/acquisition

Steps

1. Assess the system context and landscape
2. Identify the architecturally-significant requirements and decision criteria
3. Evaluate candidate technologies against quality attribute decision criteria
4. Validate architecture decisions and technology selections through focused prototyping

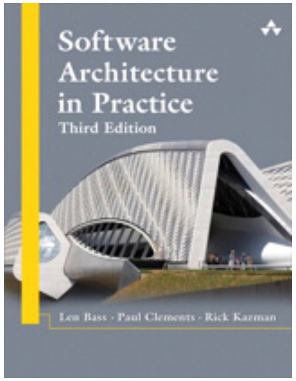


Some Example Scalability Prototypes - Cassandra



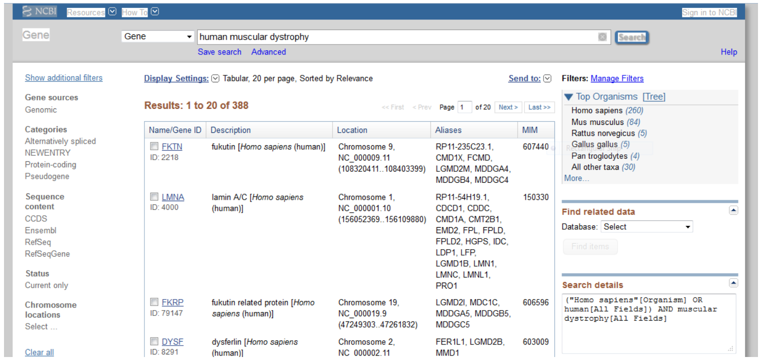
Knowledge Capture and Dissemination

in Software Engineering

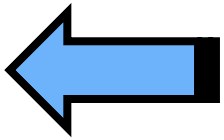


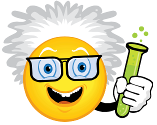
Johannes Gutenberg, circa 1450

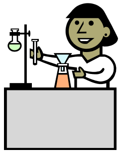
in Science (e.g. biology - http://www.ncbi.nlm.nih.gov)



Submission
Validation
Curation







Software Engineering Institute | Carnegie Mellon University

Software Architecture:
Trends and New Directions
#SEIswArch

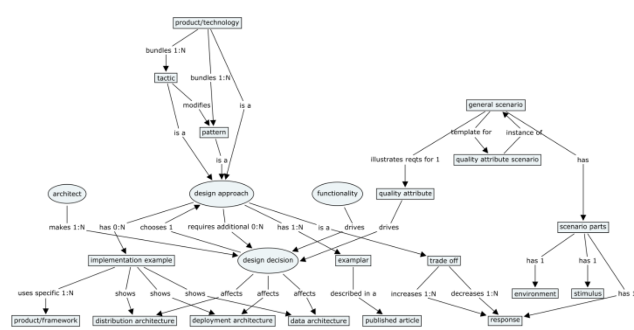
QuABase –
A Knowledge Base for Big Data System Design

Semantics-based Knowledge Model

- General model of software architecture knowledge
- Populated with specific big data architecture knowledge

Dynamic, generated, and queryable content

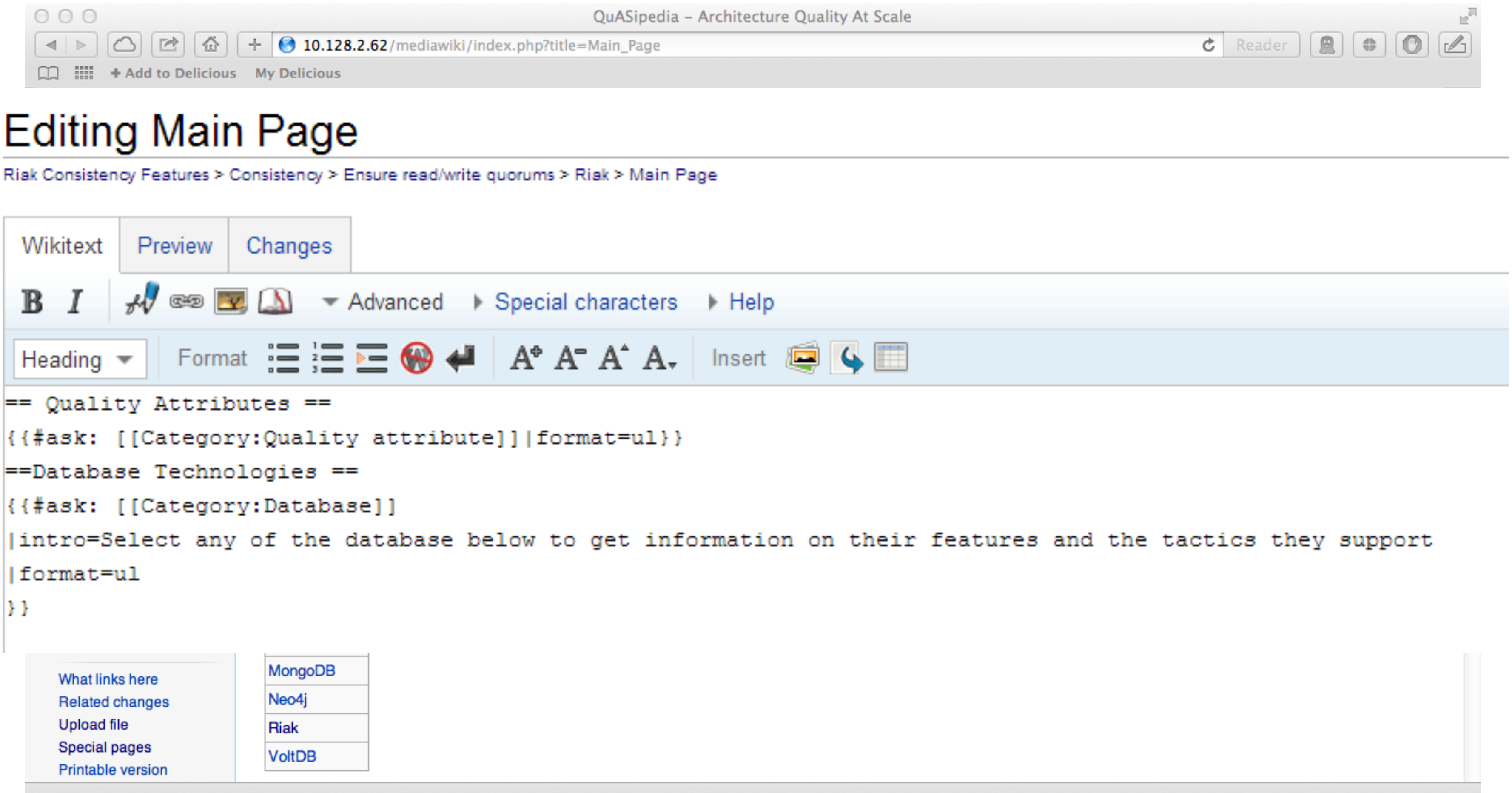
Knowledge Visualization



QuABase Demo

QuABase Demo

1



QuABase Demo

Page | Discussion | Read | Edit with form | Edit | View history | Search | Go | Search

Consistency

Ensure read/write quorums > Form:Tactic > Ensure read/write quorums > Main Page > Consistency

Description

[edit]

Consistency issues in distributed systems stem from replication and the spatial separation of data objects, and when two or more objects must be updated together to maintain logical consistency. Both these issues occur commonly in big data systems, and hence consistency is a fundamental quality attribute for big data systems.

General Scenario for Consistency

[edit]

Stimulus	A write to single data object is issued (OR) A single writer updates two or more objects to maintain consistency between them (OR) Two writers attempt to update the same object simultaneously
Environment	Distributed database with replication (OR) Non-distributed and non-replicated database (OR) Cached database access
Response	Read-after-write consistency: after a write operation on data object X the new value will always be seen by readers of X at some time in the future Updates to two or more data objects by a single writer result in consistent values across the objects through either successful updates or an error that rolls back object values to their previous state
Response Measure	Time for all object replicas to store same value after write succeeds Multiple objects updated successfully together or an error is issued and they are returned to their previous state

Quality Attribute Scenarios and Tactics for Consistency

[edit]

Quality Attribute Scenario	Tactics
Ensure eventual consistency in a replicated, distributed database	Asynchronous replica update Hinted handoffs
Ensure eventual consistency when making multiple object updates	Distributed transactions Conflict resolution
Ensure strong consistency for a write-write conflict	Conflict resolution Ensure read/write quorums Queued Writes
Ensure strong consistency in a replicated, distributed database for a single object update	Ensure read/write quorums Read from master only Write to all replicas
Ensure strong consistency in a replicated, distributed database for multiple object updates	Distributed transactions Denormalized data model
Ensure strong consistency in an unreplicated, non-distributed database for multiple object updates	Denormalization (Nested records)

Navigation
- Main page
- Recent changes
- Random page
- Help

Contribute
- Add a new quality attribute
- Add a new quality attribute scenario
- Add a new tactic

Toolbox
- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link
- Page information
- Browse properties

QuABase Demo

Ensure read/write quorums

Riak Consistency Features > Riak > Riak Consistency Features > Consistency > Ensure read/write quorums

Description

[edit]

Assuming there are N replicas of any object, a writer may specify that a quorum of the replicas must be updated before the write succeeds. This ensures that a majority of the replicas are updated before the write completes. If all writers perform quorum writes, this also prevents write-write conflicts as only one writer can ever achieve quorum at any instant.

To ensure all readers see the updated value after any write completes, readers must also specify that a quorum of object values must be the same before the read succeeds. This ensures that a reader cannot see a value at a replica that has not yet been updated with the new value.

In either case, if a quorum of replica objects cannot be written to or read from, the operation fails.

The general form or the requests to achieve strong consistency are: Qr + Qw > N Qw > N/2

A number of NoSQL databases provide quorum mechanisms for readers and writers to be able to tune consistency. This is typically specified on a per-write call to enable each write to be tuned accordingly.

Improves Quality	Consistency
Reduces Quality	Performance, Availability
Related Tactics	Hinted handoffs

Implementations

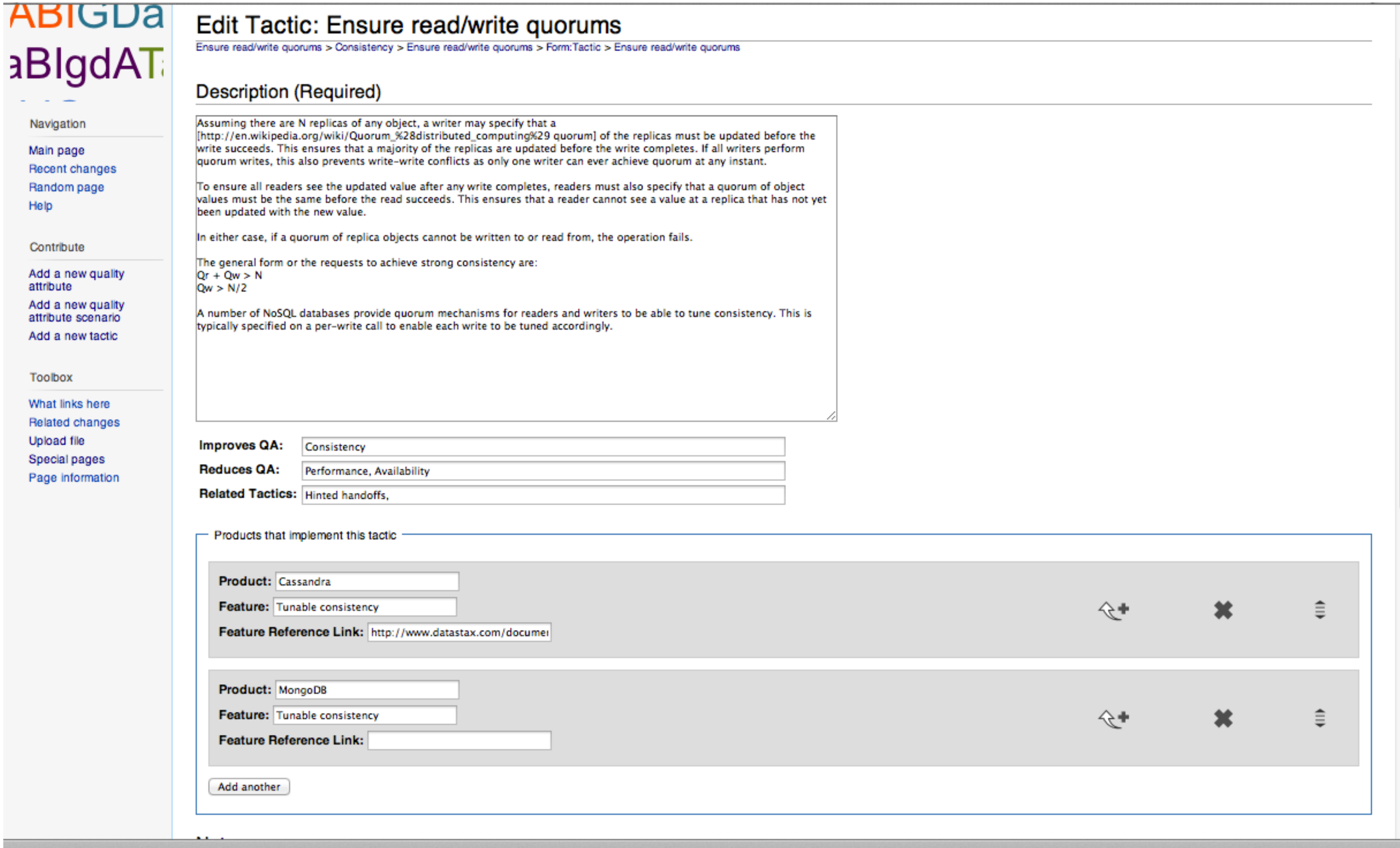
[edit]

This tactic is supported by the feature Tunable consistency of the product Cassandra.

This tactic is supported by the feature Tunable consistency of the product MongoDB.

This tactic is supported by the feature Tunable consistency of the product Riak.

QuABase Demo



QuABase Demo

Ensure read/write quorums

Description

[edit]

Assuming there are N replicas of any object, a writer may specify that a quorum of the replicas must be updated before the write succeeds. This ensures that a majority of the replicas are updated before the write completes. If all writers perform quorum writes, this also prevents write-write conflicts as only one writer can ever achieve quorum at any instant.

To ensure all readers see the updated value after any write completes, readers must also specify that a quorum of object values must be the same before the read succeeds. This ensures that a reader cannot see a value at a replica that has not yet been updated with the new value.

In either case, if a quorum of replica objects cannot be written to or read from, the operation fails.

The general form or the requests to achieve strong consistency are: Qr + Qw > N Qw > N/2

A number of NoSQL databases provide quorum mechanisms for readers and writers to be able to tune consistency. This is typically specified on a per-write call to enable each write to be tuned accordingly.

Improves Quality	Consistency
Reduces Quality	Performance, Availability
Related Tactics	Hinted handoffs

Implementations

[edit]

This tactic is supported by the feature Tunable consistency of the product Cassandra.

This tactic is supported by the feature Tunable consistency of the product MongoDB.

This tactic is supported by the feature Tunable consistency of the product Riak.

QuABase Demo

Riak

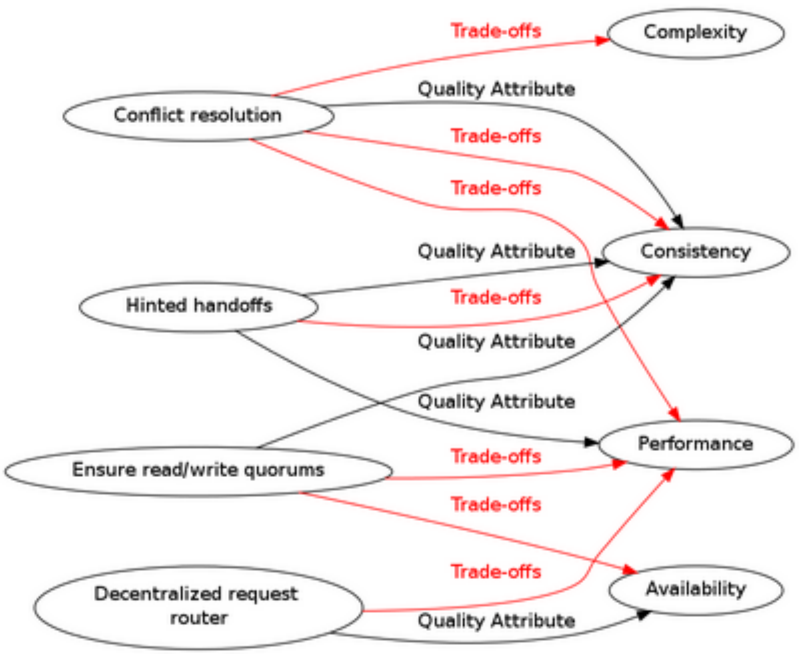
Main Page • Riak • Template:Database Main • Main Page • Riak

Overview	cool Key value db
Model	Key-Value

Riak Features

Feature Category
Riak Consistency Features

Tactics Supported by Riak



black: Quality Attribute
red: Trade-offs

QuABase Demo

Riak Consistency Features

Riak > Riak Consistency Features > Riak > Riak Consistency Features

LEAP4BD

Evaluation

Features

Database	Riak

Object-Level isolation on updates	supported
ACID transactions in single database	not supported
Distributed ACID transactions	not supported
Specify Quorum Reads/Writes	in client

Specify number of replicas to write to	in client
Behaviour when write cannot complete on specified number of replicas	no rollback: write returns replication error
Writes configured to never fail	supported
Specify number of replicas to read from	in client
Read from replica master only	not supported
Updates applied to transaction log before returning from write	supported
Object level timestamps to detect conflicts	supported
Efficient protocol to rapidly propagate updates across replicas (minimize inconsistency window)	by default

add explanations here

Categories: Consistency Features | Strong Consistency | Eventual Consistency

Status

LEAP4BD

- Initial trial with DoD client near completion
- Rolling out as an SEI service

QuABase

- Design/development in progress
- Validation/testing over summer

Software Engineering for Big Data Course (1 day) and tutorial (1/2 day)

- SATURN 2014 in Portland, May 2014
  - http://www.sei.cmu.edu/saturn/2014/courses/
- WICSA in Sydney, Australia April 2014
- Both available on request

Thank you!

http://blog.sei.cmu.edu/

OCT 21 2013 — Addressing the Software Engineering Challenges of Big Data

JAN 13 2014 — The Importance of Software Architecture in Big Data Systems


Browse Early Access Articles > Software, IEEE > Volume:PP Issue:99

Distribution, Data, Deployment: Software Architecture Convergence in Big Data Systems

Full Text as PDF

2 Author(s) — Gorton, I. ; CMU, Pittsburgh ; Klein, J.

Abstract | Authors | References | Cited By | Keywords | Metrics

Download Citations · Email · Print · Request Permissions · Save to Project

Exponential data growth from the Internet, low cost sensors, and high fidelity instruments has fueled the development of advanced analytics operating on vast data repositories. These analytics bring business benefits ranging from web content personalization to predictive maintenance of aircraft components. To construct the data repositories that underpin these systems, there has been rapid innovation in distributed data management technologies, employing schema-less data models and relaxing consistency guarantees to satisfy scalability and availability requirements. This paper describes the challenges of these "big data" systems that confront software architects. We show how distributed software architecture quality attributes are tightly linked to the both the data and deployment architectures. This causes a consolidation of concerns, and designs must be closely harmonized across these three structures to satisfy quality requirements.

This document is available in the event console materials widget