Can Blockchain Technology Play a Role in iRODS?

Arcot (Raja) Rajasekar
rajasekar@unc.edu
The University of North Carolina at Chapel Hill

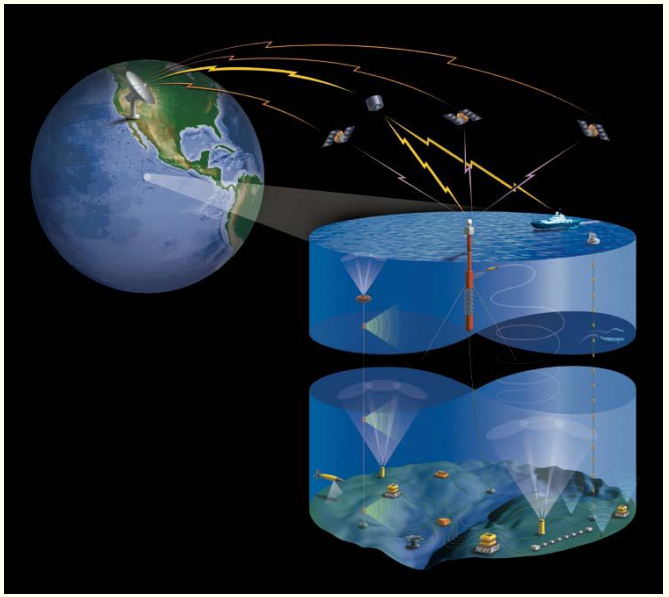
THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

D·I·C·E

UNC SCHOOL OF INFORMATION AND LIBRARY SCIENCE

Outline

- Block Chain Technology
- Related Functionalities in iRODS
- Looking Ahead:
  - Applying Block Chain Technology in iRODS
- Q&A



Block Chain Technology

What is Block Chain Technology?

- BCT, at its core, is a distributed transaction recording system with no centralized control.
  - View it as a distributed storage for digital assets
  - In a sense similar to iRODS
- It's a digital ledger which provides some important functionalities:
  - Tamper proof (Immutability)
  - Anybody can see the data but cannot corrupt it
  - Decentralized Governance
  - Highly secure
  - Provenance (time-stamping)
  - Digital Signatures for Ownership
  - Anonymity (if needed)
  - Programmable (ala triggers)

From https://www.cyberbahnit.com/

Applications of BCT

- Crypto-currencies
  - the most known application of BCT (e.g. BitCoins)
  - Distributed Ledger for holding and verifying transactions
    - New mined coins or transfers of old coins
    - Miners are also verifiers (needs verifiers)
  - Broadcast of transactions (contracts) across a P2P network
  - Validation using 'known' cryptographic algorithms
  - Multiple verifiers – consensus (helps integrity) (rewards)
  - Verified transactions added to blocks, chained, timestamped, encrypted and distributed – makes it immutable
  - Concept of Wallet: Public key and Private key combination
    - Used as a validation of a user
    - Digital signature
    - Anonymity – But can be stolen

From https://www.reuters.com/

Other Applications of BCT

- Secure sharing of documents
  - Medical, financial, …
- NFT (non-fungible tokens)
- Supply chain and logistics monitoring
- Real estate transaction processing
- Voting
- Money transfers using Cryptocurrencies
- Security for Real-time IoT operations
- Government operations (replacing databases)
- Royalties and Patents

Internals of Block Chain Technology

- Nodes
  - Maintain copies of transactions or hash value of transaction
- Ledger
  - database
  - Three types
    - Public Ledger
    - Distributed Ledge
    - Decentralized ledger
- Nonce
  - stands for “number only used once”
  - a unique random 32-bit a number added to a hashed or encrypted block in a blockchain.
- Hash
  - data is mapped to a fixed size using hashing
  - hash value of one transaction is the input of another transaction
- Triggers
- Wallet (client side)
  - Public Key Private Key
  - Authentication
  - Balance – in case of crypto currency

From: https://sciencenotes.org/steps-scientific-method/

Internals of iRODS technology

- You all should know what they are?

From: https://sciencenotes.org/steps-scientific-method/

iRODS and BCT

- Distributed Nodes
- Ledger
- Transactions
- Verification
- Triggers
- Wallet
- Blocks
- Hash and Chaining

- Ledger distribution

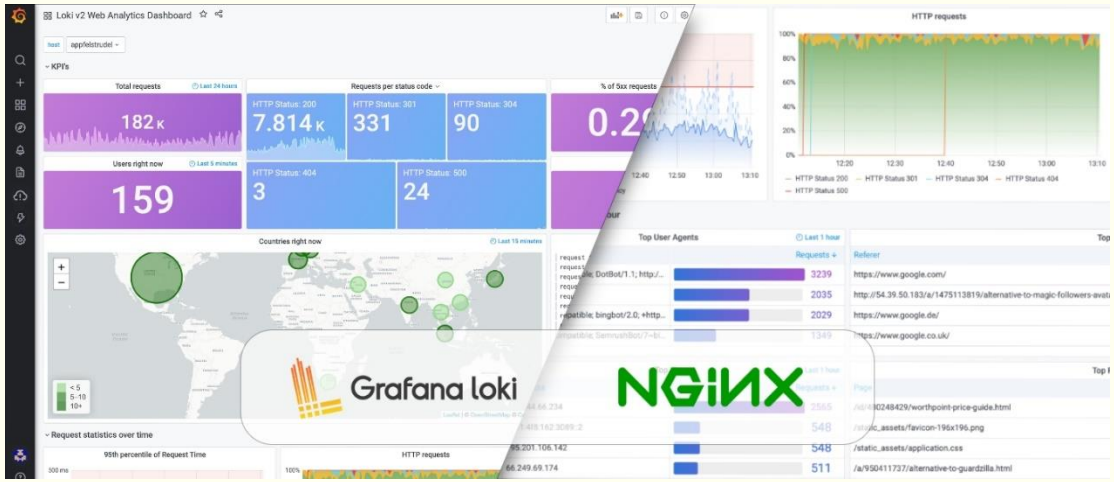
- Distributed Resources
- Metadata Catalog
- Audit Trail & Server Log
- Access Control
- Rules
- Users

- Data & Collection
- Replication

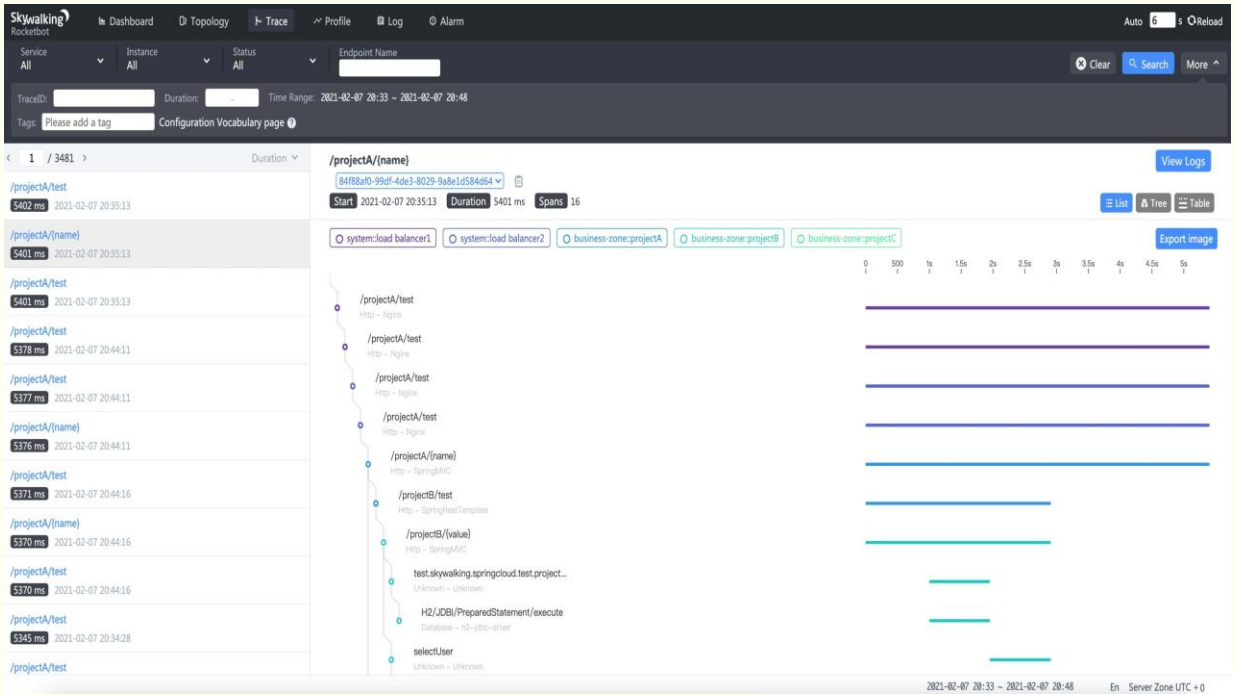
Example Observability Systems

- **DataStax**

- **Grafna Dashboard**

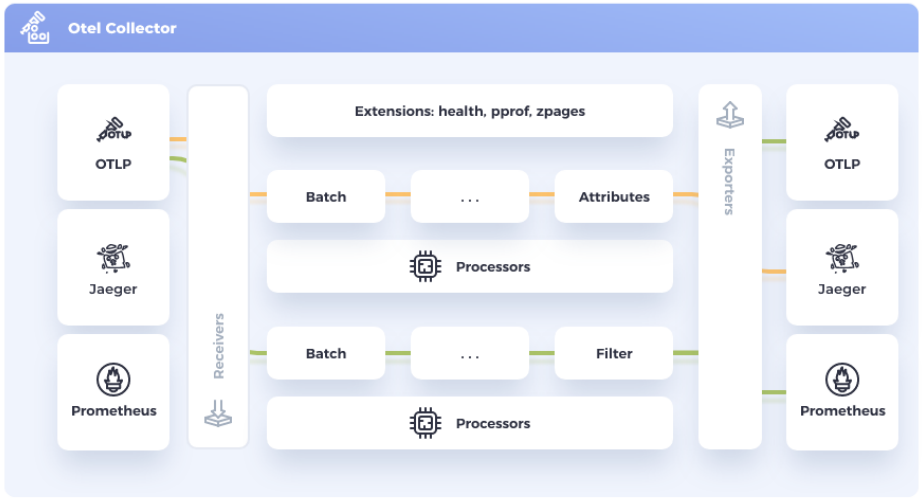


Example Observability Systems



← Apache Skywalker

Open Telemetry →



Three Pillars of Observability

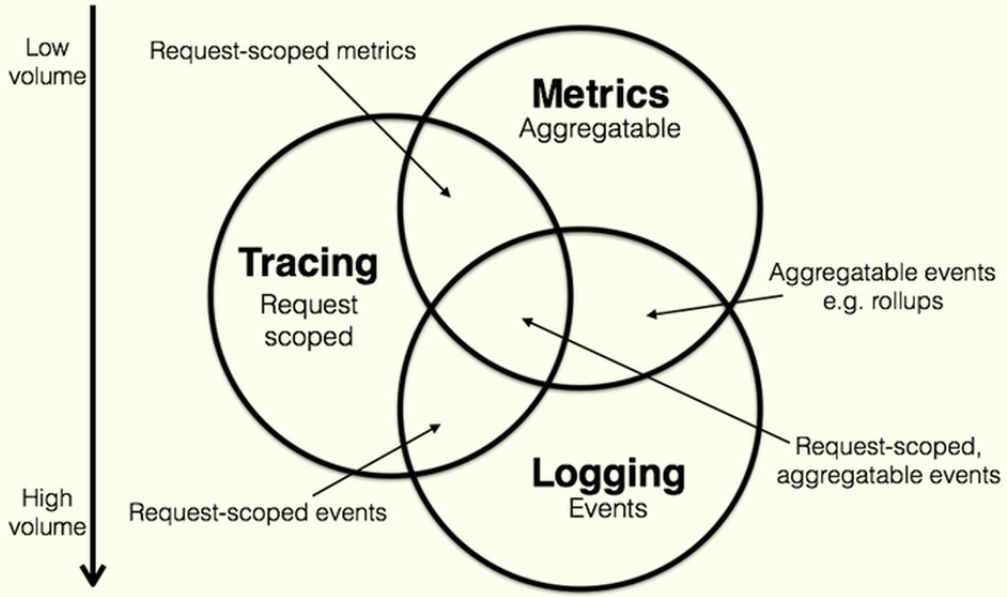
- Logging: collects information about events happening in the system and helps find unexpected behavior
- Tracing: collects information to create an end-to-end view of how transactions are executed in a distributed system. Tracing can recognize a problem through comparing and contrasting.
- Metrics: provide a real-time indication of how the system is running. Metrics can be leveraged to build alerts, allowing proactive reaction to unexpected values

From: https://www.humio.com/

Two More Pillars:

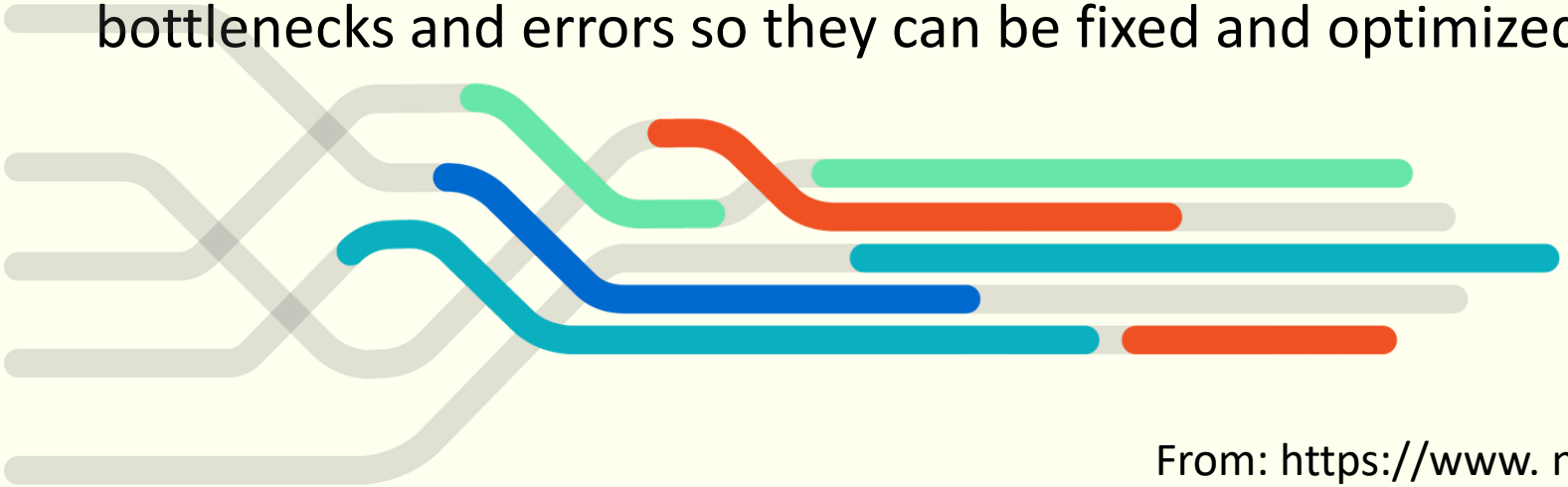
Visualization: Visual Cues for abnormalities

Analytics: Deep analytics to predict faults, failures and service degradation



Journey: A User Experience

- Tracings create an end-to-end view of how transactions are executed in a distributed system. They also capture end-to-end and inter-service latencies of individual calls in a distributed journey
- Journey: The sum total of all activities a user performs during a session. A journey can have multiple sub-journeys. Each journey can be made of several paths which can be parallel in a distributed system.
- A journey captures timings, possibly call and return expressions, status code and anything else that an Observer deems to be necessary.
- Journey can be abstracted into templates and help find bottlenecks and errors so they can be fixed and optimized.

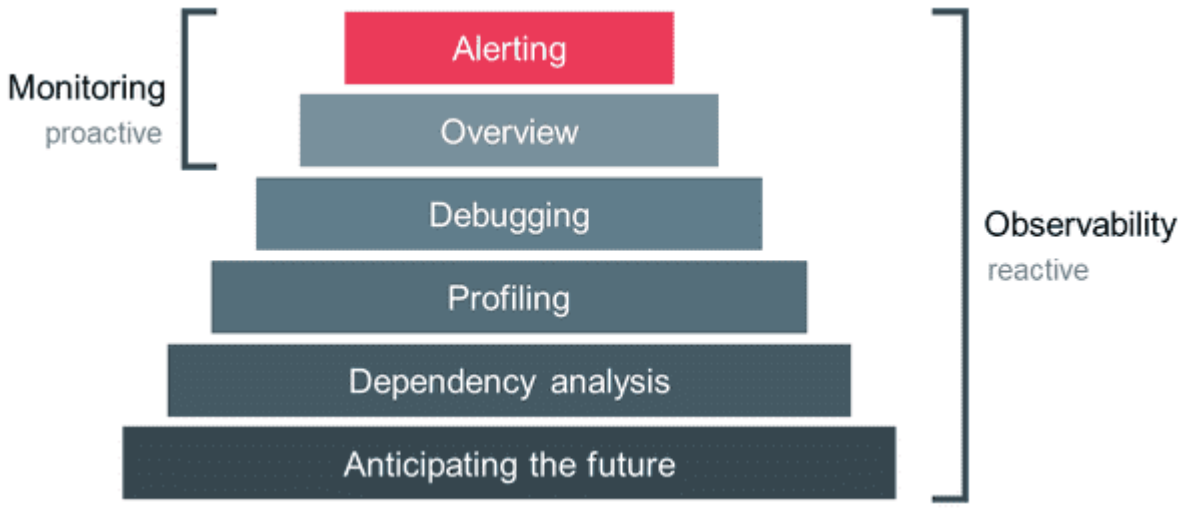


From: https://www. newrelic.com/

Observability in iRODS: Current Status

- Server Logs: collects information about system events and error messages happening in the system. Can be used to find unexpected behavior (distributed)
- Audit Trails: collects user-defined information on triggered action. Can be used to recreate traces that are executed across distributed iRODS servers (centralized).
- Status Metadata: Can store persistent information that can help for further metrics (centralized)

iRODS is currently supportive more towards Monitoring activities than towards Observability.



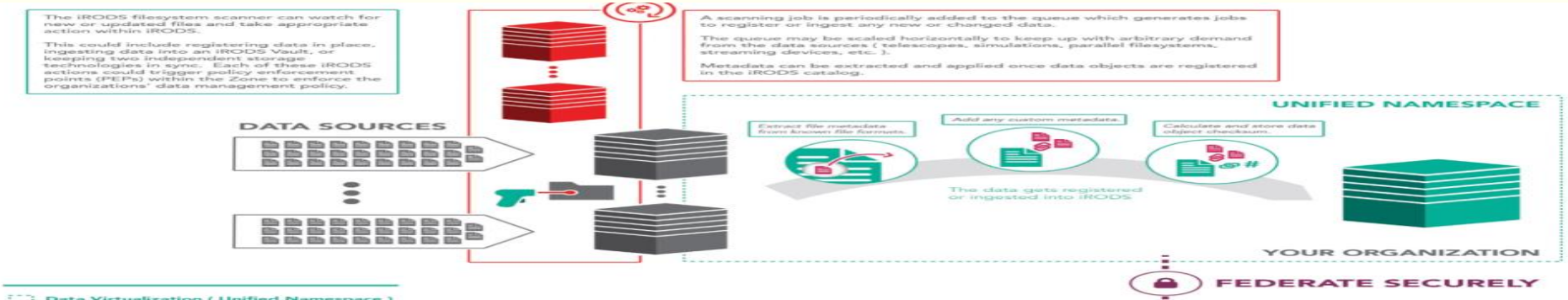
From: https://ish-ar.io/observability/

Observability in iRODS

- Towards better performance with proactive metrics & analysis:
  - Help iRODS become better and more pro-active in maintaining performance
  - Help systems that use iRODS to apply iRODS observability metrics to become better and pro-active in maintaining performance
- Server Logs, Audit Trails and Status Metadata in iRODS provide a strong and stable foundation for performing Observability.
- Use of policies, rules and microservices provide one more level for gaining information to perform observability
- Missing: Metrics, Journeys, Visualization and Analytics

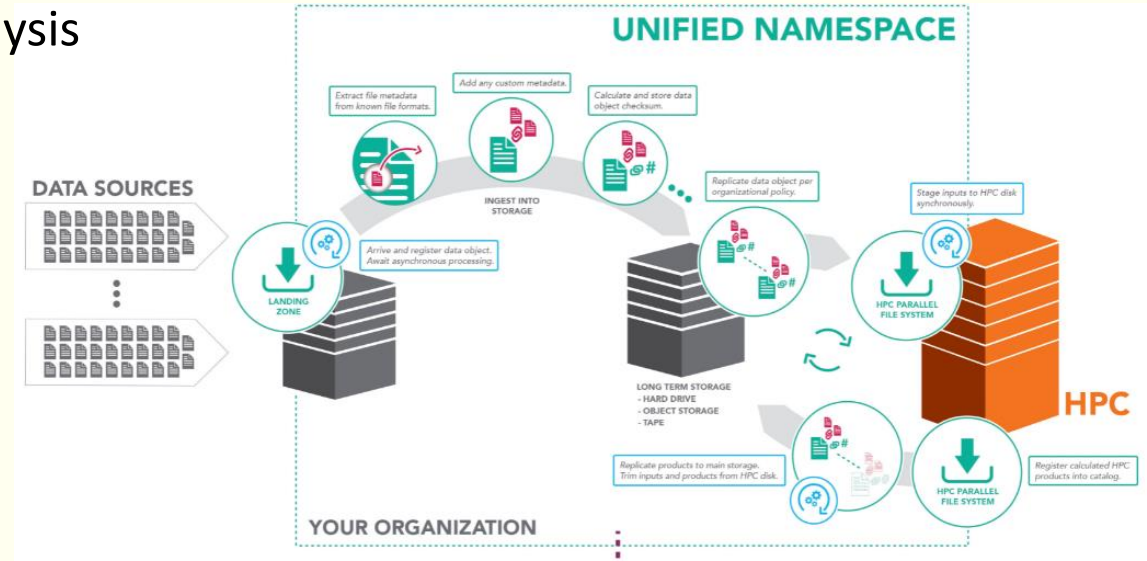
iRODS Observability: Metrics

- Application Performance Monitoring (APM): To check whether the system satisfies the SLA contracts, meets performance standards, identify bugs and potential issues, and provide flawless user experiences via close monitoring of IT resources.
- Reduce MTTR (Mean Time To Resolution)
- Continuous Monitoring towards Proactive Remediation
- Alerts and Simple Analysis
- Metrics: What can we monitor in iRODS (not a comprehensive list)
  - CPU/Memory Usage
  - Network Traffic
  - Database Load
  - Error Types/Rates
  - Request rates
  - Response times (mean, max, min)
  - Bandwidth/Throughput
  - Concurrent Connections
  - Number of instances/threads
  - Microservice/function usage/time
  - Uptime, Restarts & Availability
  - User Experience (happy faces)
  - Other Software KPIs



iRODS Observability: Journeys

- Distributed Tracing (DT): Chaining of services and peer-to-peer connections across distributed systems makes it hard to trace the activities of a session but is critical for performance monitoring.
- DT helps identify bottlenecks across dynamic and heterogenous infrastructures
- Journeys: Session level performance analysis and monitoring
  - Distributed Transaction Monitoring and Analysis
  - Create User or Application Profiles
  - Define Patterns and Templates of Journeys and Sub-journeys
  - Latency optimization
  - Failure Models – Alternate Pathways
  - Service Dependency Analysis
  - Critical Path Analysis
  - Root Cause Analysis



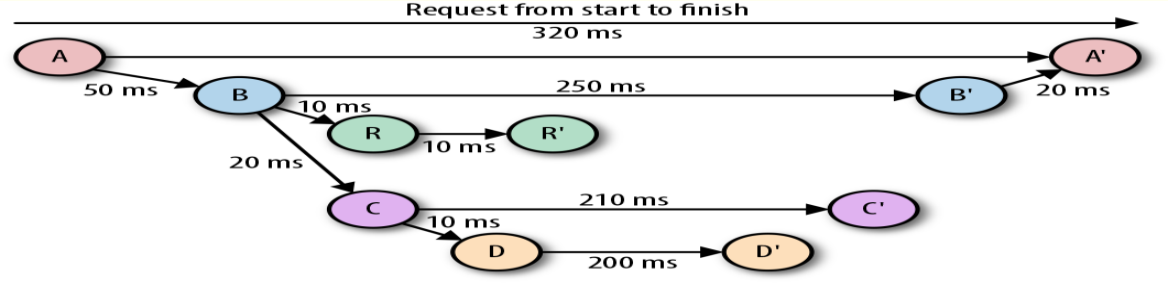
iRODS Observability: Analytics

- Predictive Analytics: What is likely to happen?
- Descriptive & Diagnostic Analytics: What happened and why it happened?
- Prescriptive Analytics: How can we avoid that happening?

Some Examples

- Statistical Analytics: Analyze metrics data for informative nuggets. Max, Min, Median, Mean, StdDev, etc. provide insights. Can be used to define norms, SLAs and expected outcomes and latencies
- Graph Analytics: Use traces and journeys to find patterns. Pattern analysis. Critical nodes and Most used nodes. Candidates for improvements. Pre-staging and pre-processing options.
- Text Analytics: Contextual data of journey to define dynamic slicing and define repeatable experiences.
- Machine Learning: Learn good and bad patterns. Successful journeys and failed journeys.
- …

We are just scratching the surface



From: https://www.oreilly.com/

iRODS Observability: Visualization

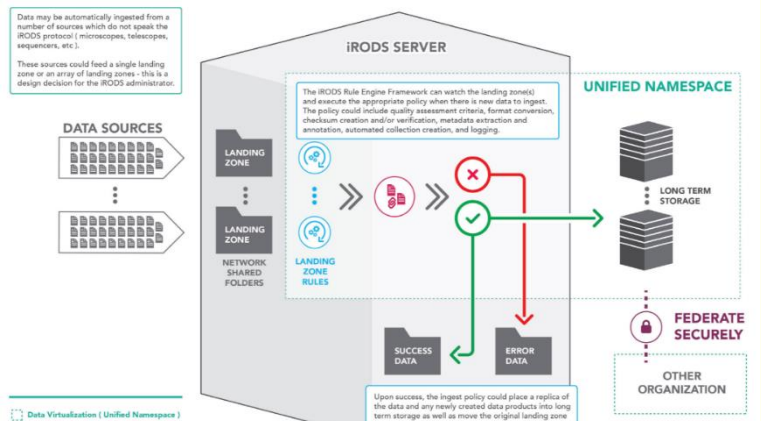
A System Administrator's Dream

From: https://www.oreilly.com/

iRODS and Observability

- Observability is becoming important because of complexities of the applications as well as need for high availability and throughput by the user community
- Observability can be used as a means to monitor the system continuously and, if possible, correct them on the fly
- Observability can also provide insight to developers on how performance can be improved
- Observability in iRODS
  - Multiple assets already available in iRODS: server logs, audit trails, metadata
  - Other assets we haven't leveraged yet: policies, rules, micro-services
- There is a clear need for Observability in iRODS
  - Metrics can be improved
  - Journeys can help in making user experience better
  - Analytics can help find problems before they occur
  - Visualization can help developers and administrators with visual cues and human analytics
- Good idea to think about when we already do enterprise level applications

Scalable, reliable resilience needs better Managed Adaptive System Support





Observability & iRODS

Q & A

rajasekar@unc.edu