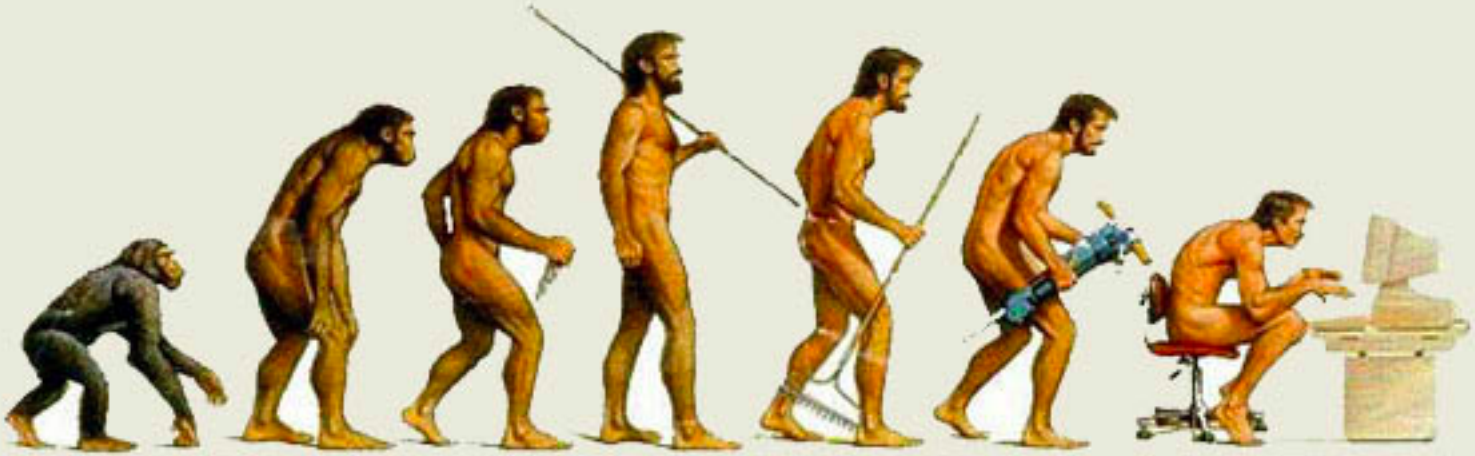
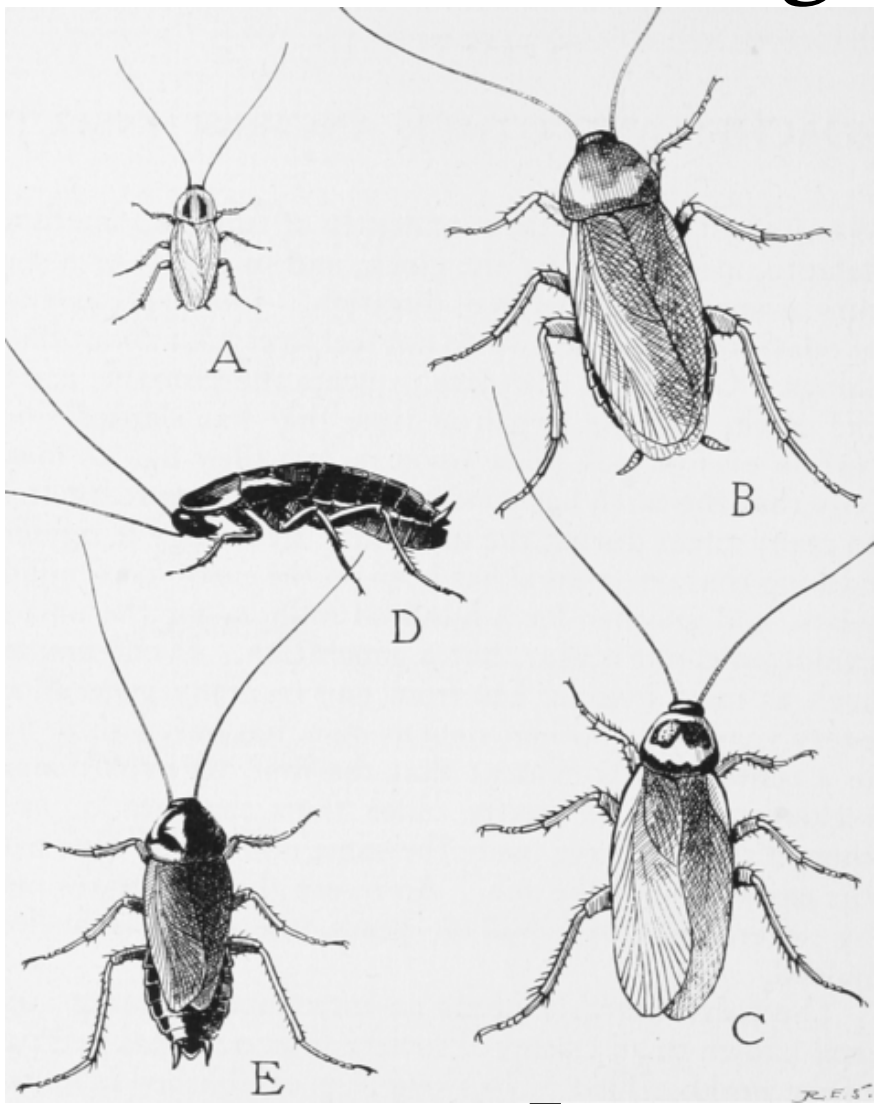
The evolution of integration

QCon London 2012

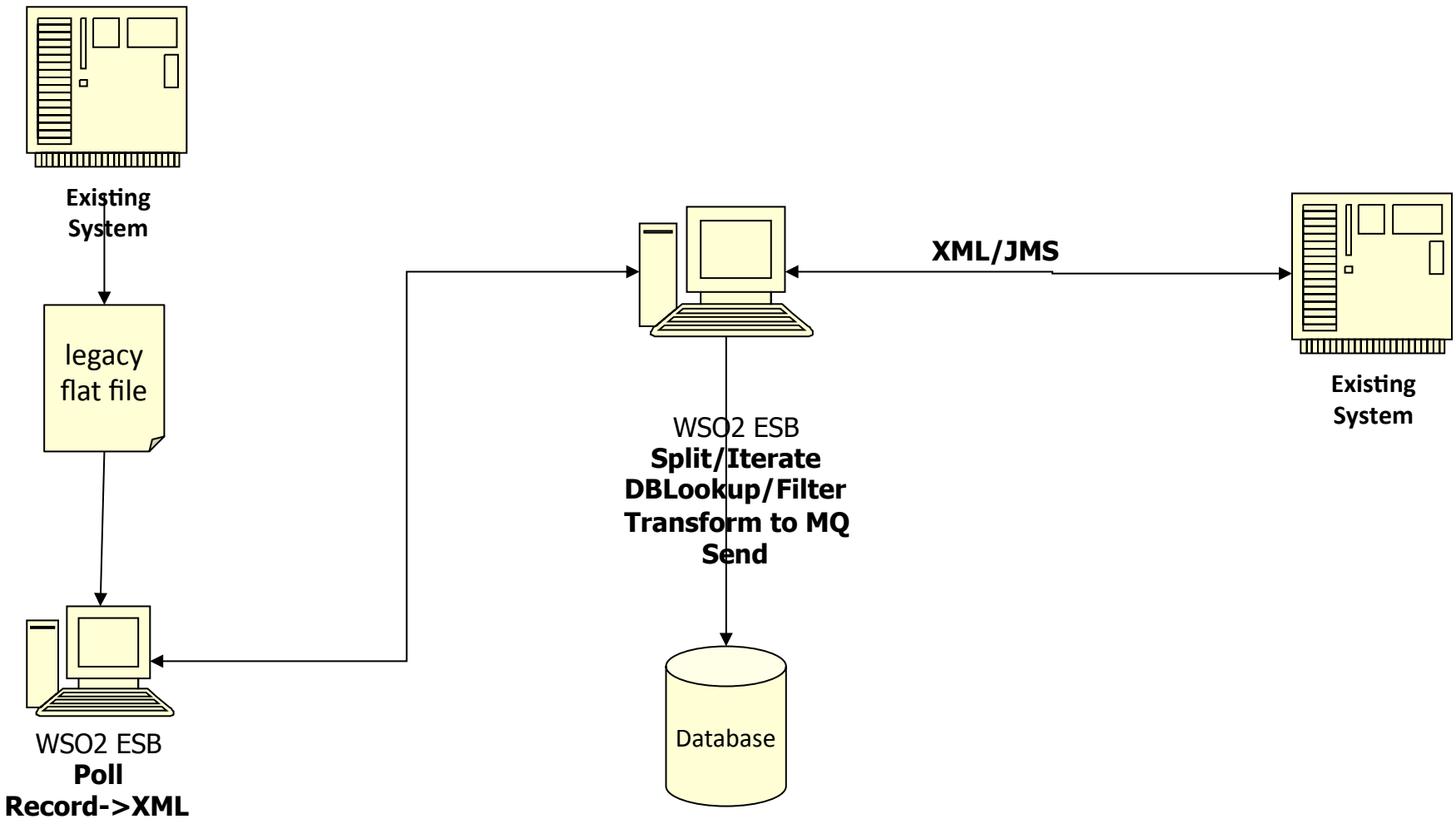
Paul Fremantle
CTO and Co-Founder, WSO2
paul@wso2.com



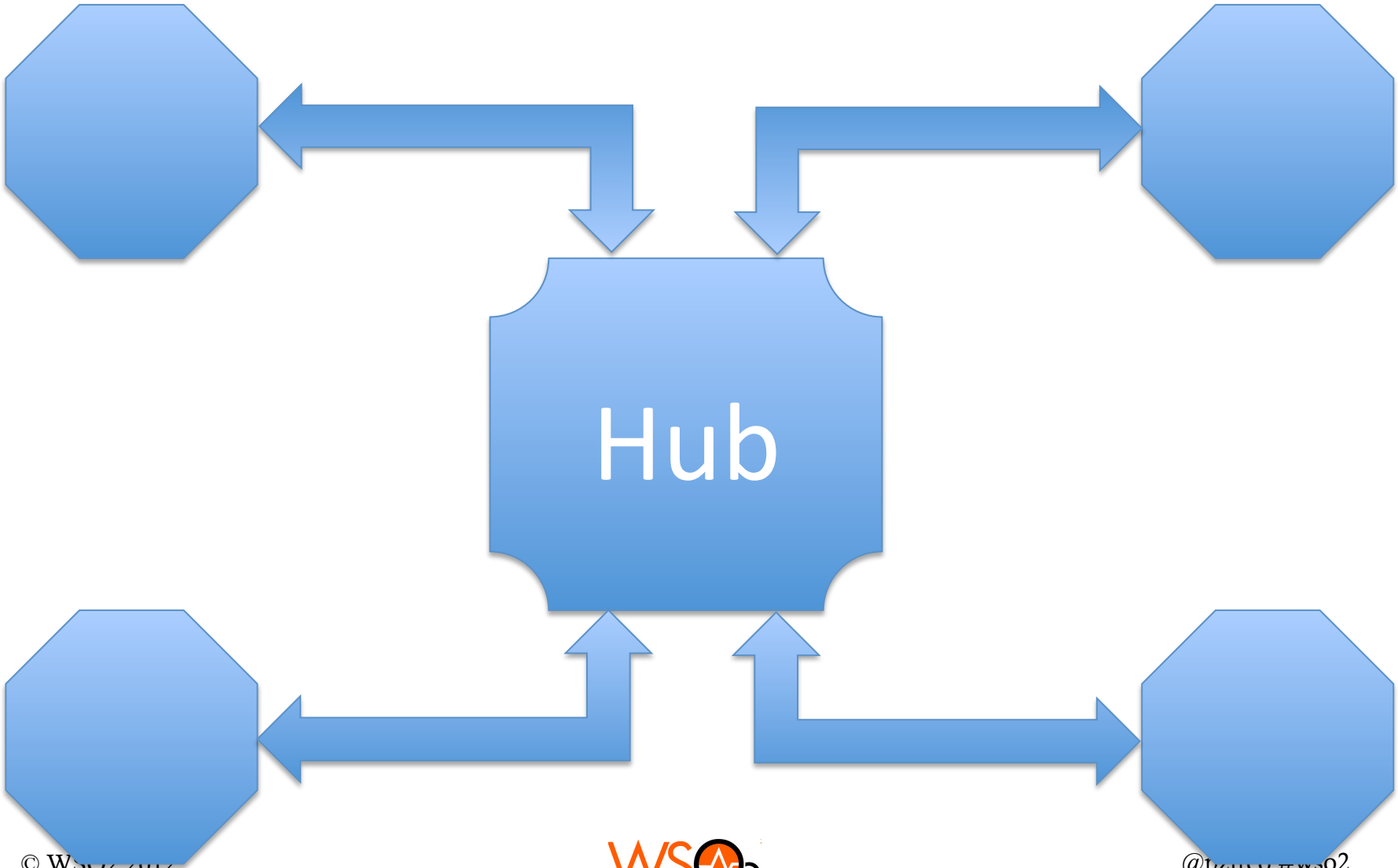
Batch file transfer - the cockroach of integration



File transfer lives on



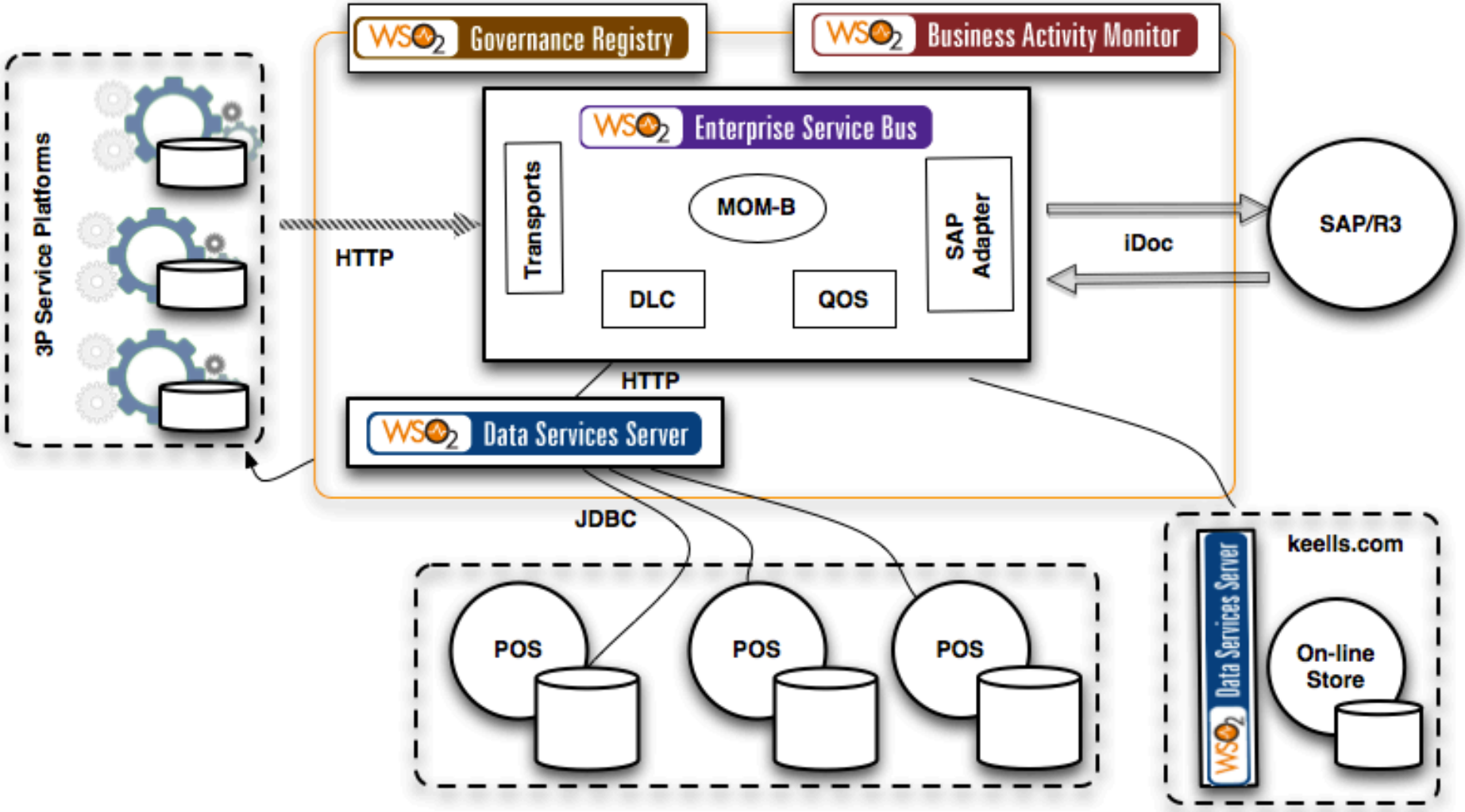
EAI



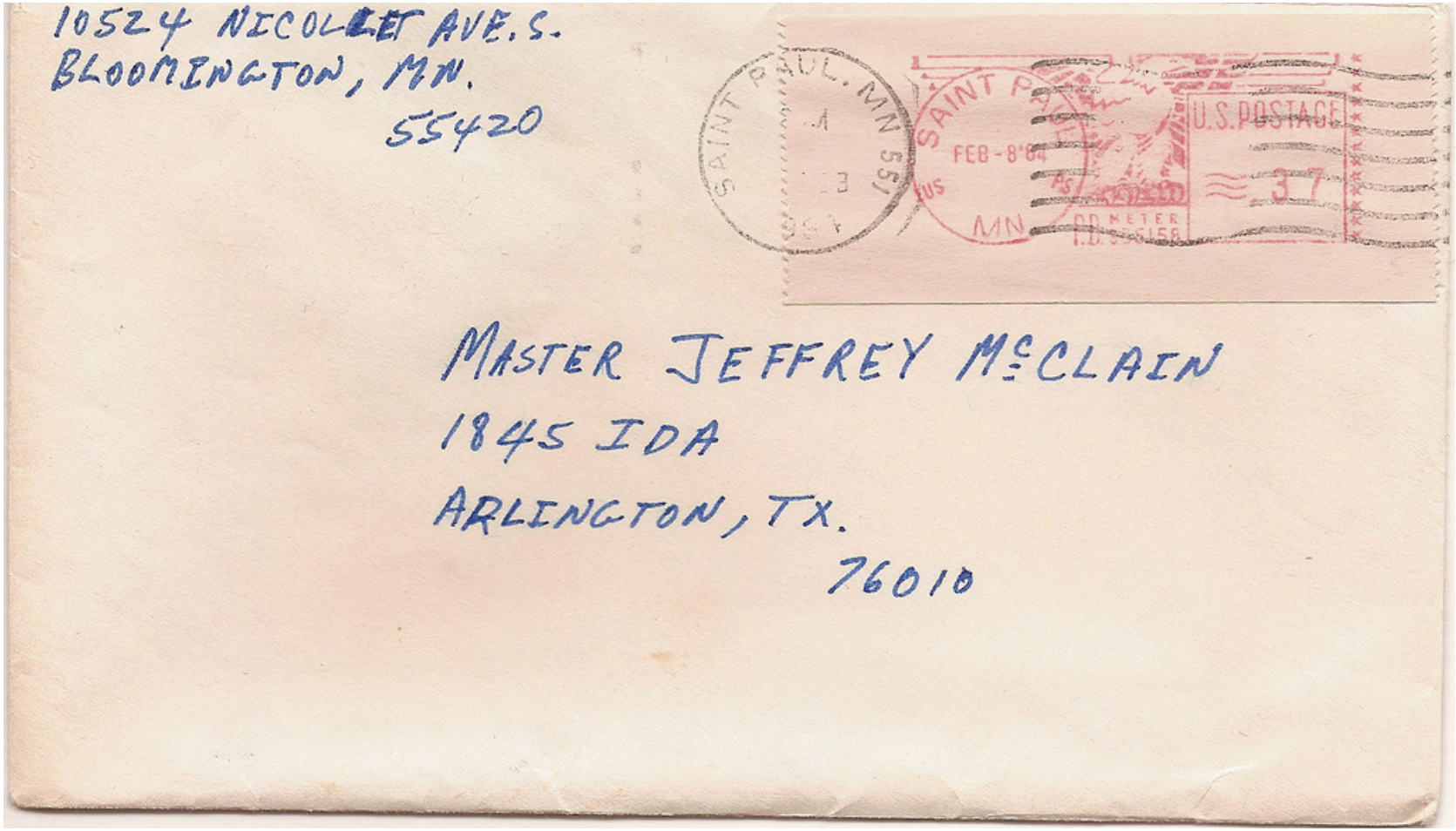
EAI hub

- Many integration models still uses the “hub model” today (even with an ESB)
  - Most vendors renamed their hub to ESB
- Why?
  - Well understood pattern
  - Easy to manage
- Why not?
  - Too many meetings with the “EAI Hub Team”
  - MQSI experiences

Hub approach with an ESB



“Message Oriented Middleware”

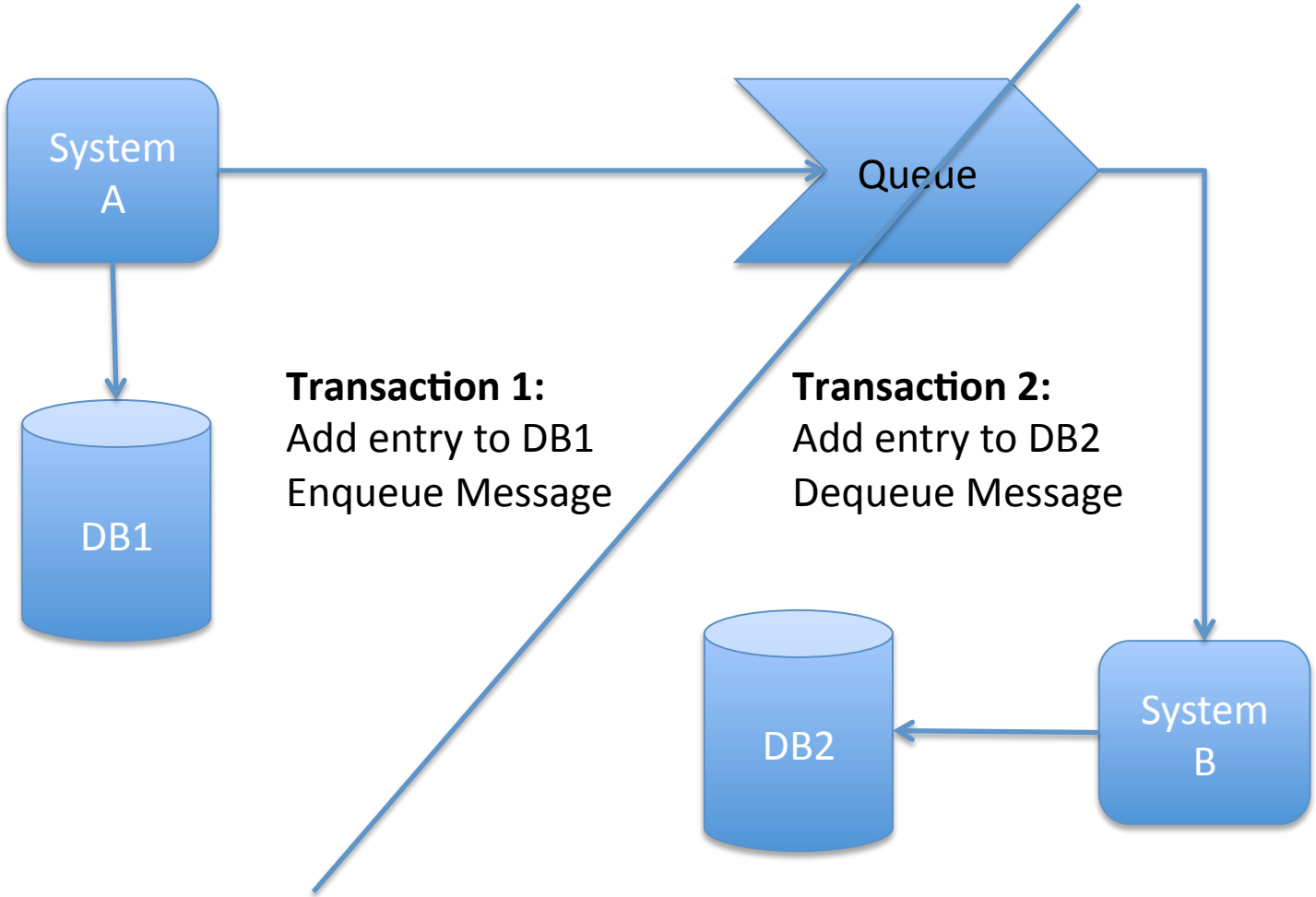


http://www.flickr.com/photos/robeast/

MOM model

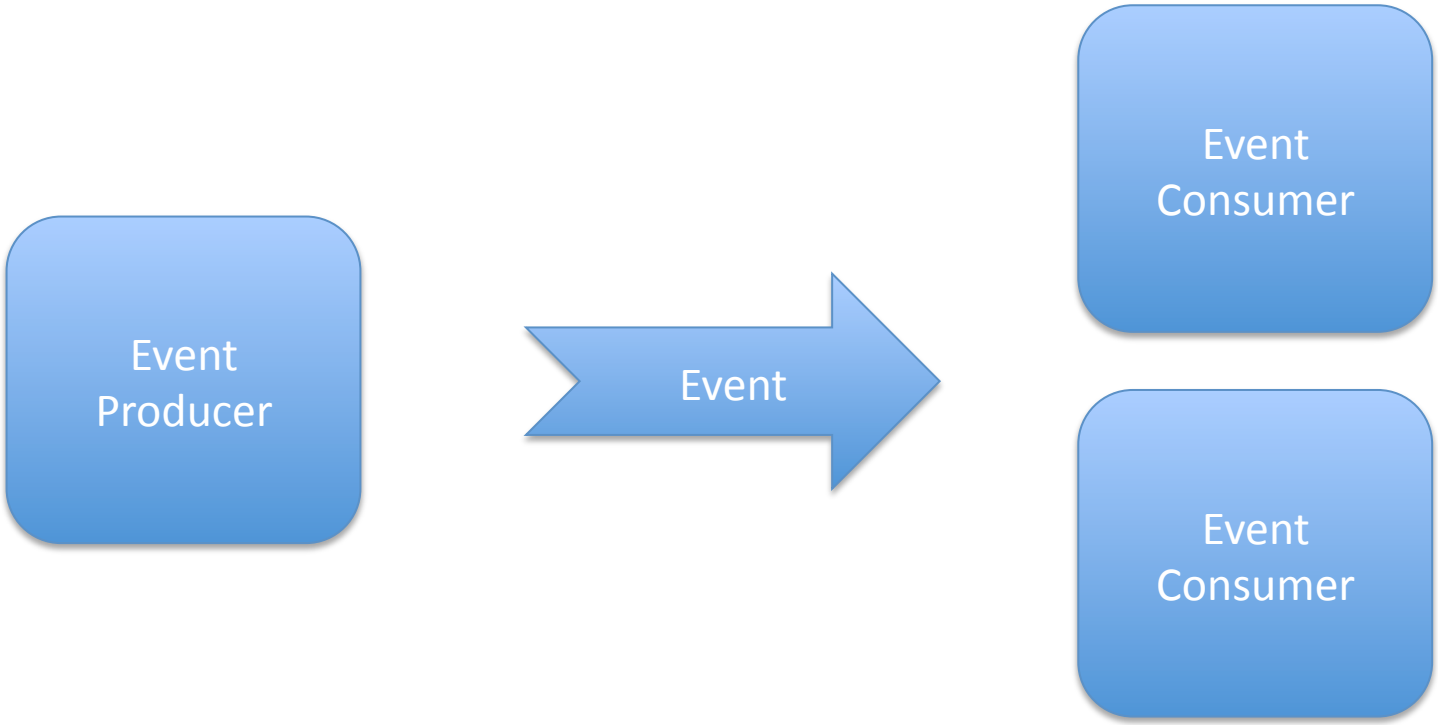
- Decouple message producers from consumers
  - Decoupled in addressing *and* in time
- Not inherently decoupled in message format
  - Though in many cases that too
- One-way asynchronous messages
  - But request-reply possible using “Reply Queues”
- Usually used with reliable delivery

Queued Transaction Processing



AMQP

Event Driven Architecture

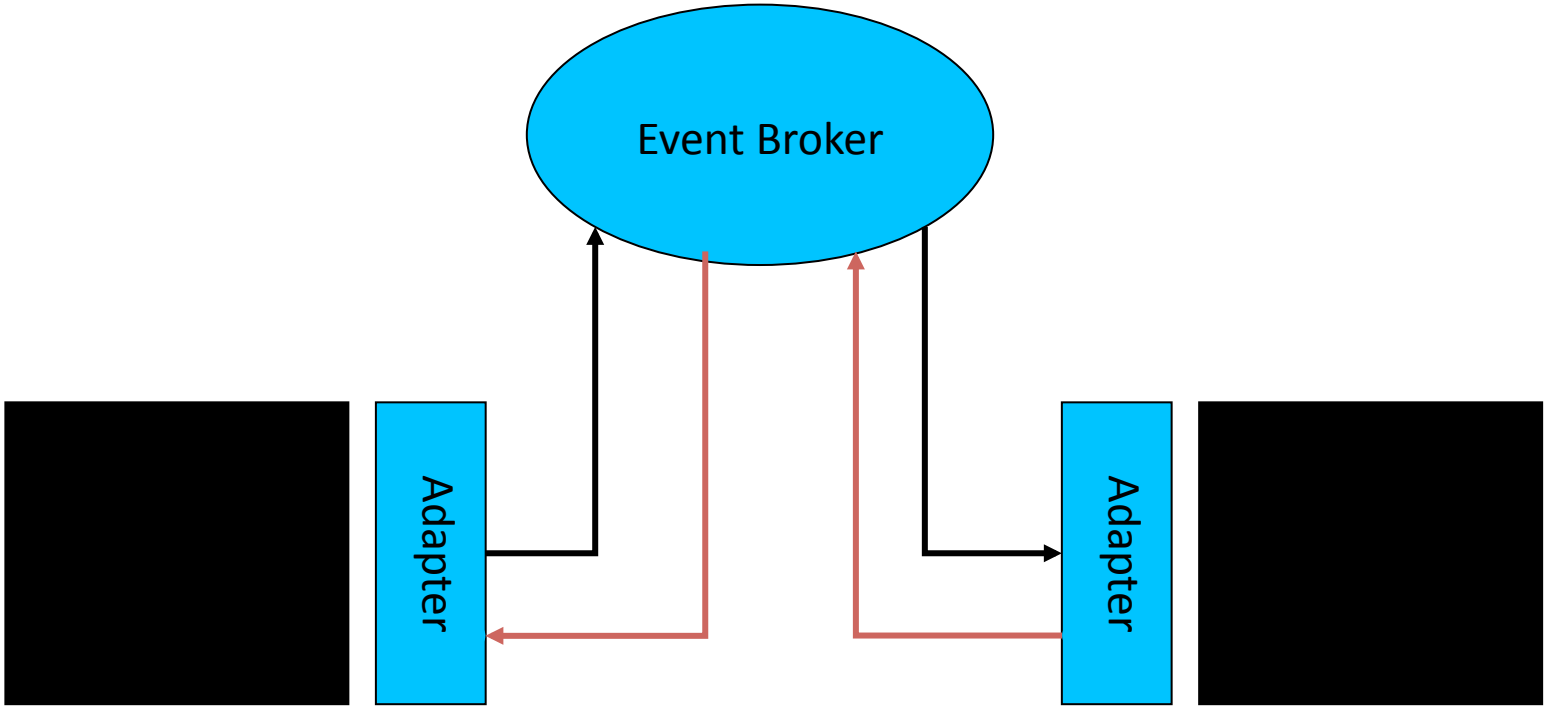


“Event Driven Architecture”

- Actually how Apache (and WSO2) work(s)
  - Mailing lists = topics
- Can be layered with reliable delivery
- Used a lot in high-volume logging, trading environments, fraud detection, etc
- *Requires a very different mindset*

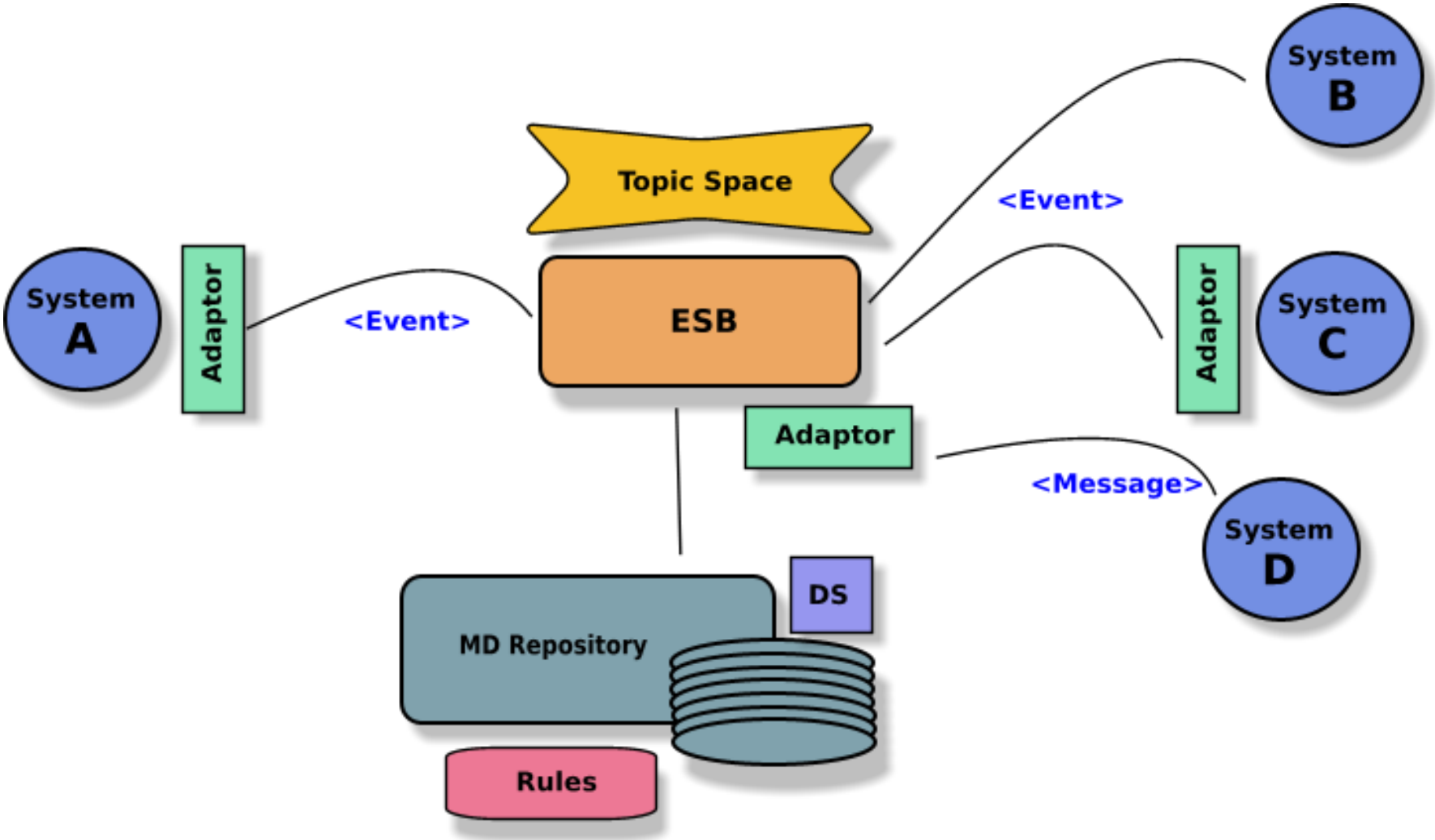
EDA

Feedback loops



http://pzf.fremantle.org/2008/09/interesting-problem-in-event-driven.html

Solution



Why did SOA evolve?

- Directly came out of XML
  - Understanding the schema and structure of messages
  - Especially within the "fabric" not just at the endpoints
- What's different?
  - Metadata
  - Policies
  - Security

Service

http://www.flickr.com/photos/yjv/

SOA failures

One consumer per service

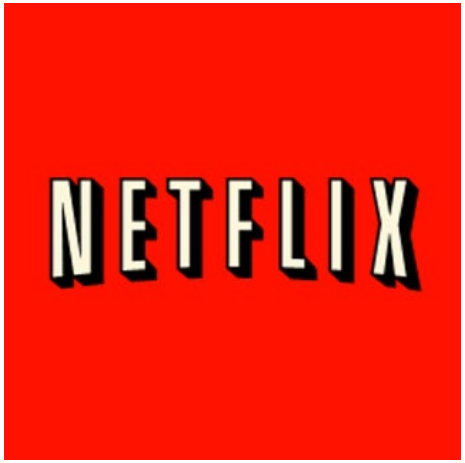
SOA failures

"Just buy an ESB from me"

SOA failures

Vendor Driven Architecture

“soa” successes



eBay Open Source

https://www.ebayopensource.org/index.php/Turmeric/HomePage

WSO2, Inc. Mail - In | Offline WSO2 Mail | WSO2, Inc. - Calend | Fremantle Family Ma | Shorten with bit.ly | Gmail - Inbox (720) | HP Photosmart C450

eBay Open Source

Search

Home | Projects | Downloads | Source | Issues | Documentation | Forums | Builds | Login/Register

Turmeric

- Home
- Get Involved
- Downloads
  - Prior Releases
- License
- Nightly Builds
- FAQ
- Documentation
  - 1.1.0
  - Overview
  - Architecture
  - Installation
  - Tutorials

Print

Fork me on GitHub

Turmeric

What is Turmeric?

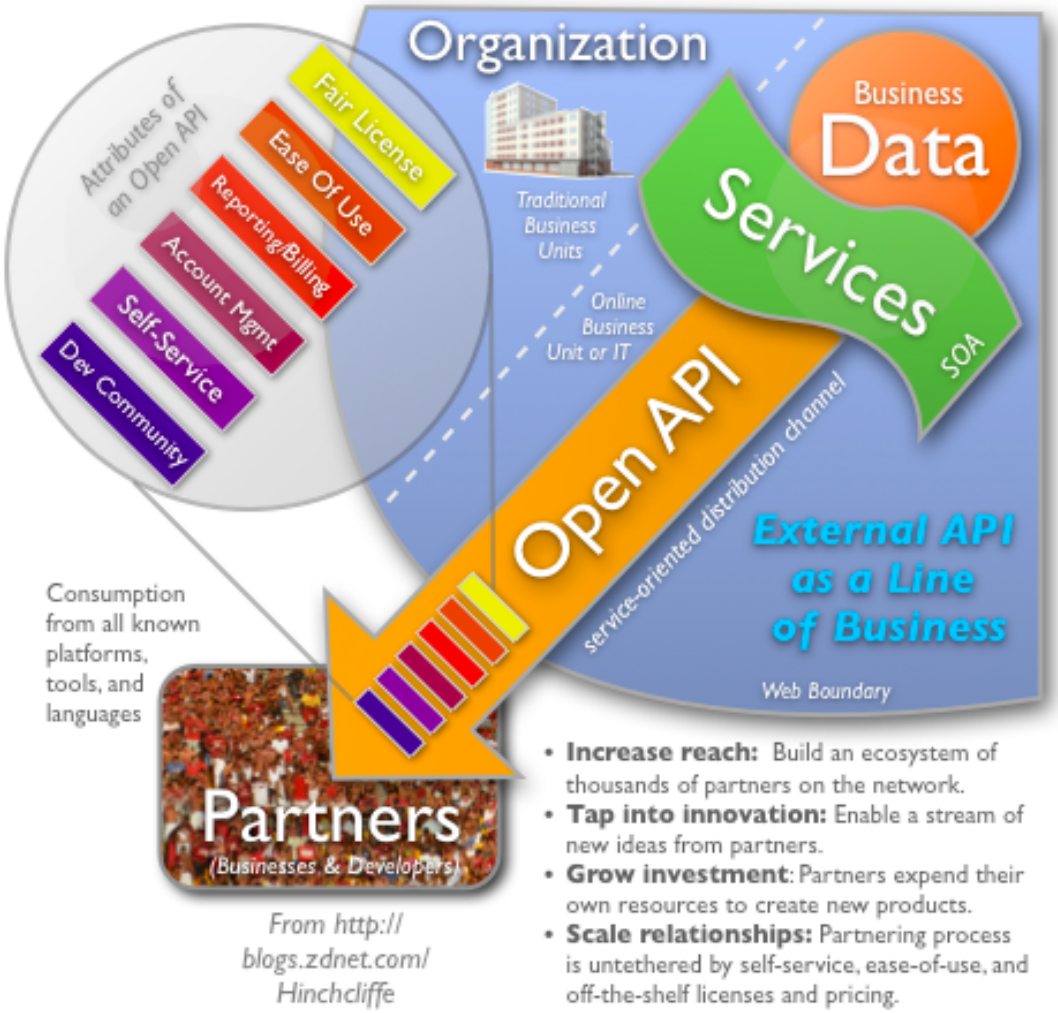
Turmeric is a comprehensive, policy-driven SOA platform that you can use to develop, deploy, secure, run and monitor SOA services and consumers. It is a Java based platform, follows the standards (SOAP, XML, JSON, XACML, etc.), and supports WSDL (SOAP style - Doc Lit wrapped mode and REST style). It supports a variety of protocols and data formats. Eclipse plugins help with the development of services and consumers. Other important features include:

- Various Quality of Service (QoS) features such as authentication, authorization, and rate limiting, which you control by defining respective policies.
- Monitoring capabilities.

Why ESB/SOA model isn't just EAI

- Policy based
  - XACML, Throttling Policy, etc
  - eBay's Internal Service Router
- Independent management
  - Loose coupling of configuration
  - Hot deploy / re-deploy / continuous delivery
- Governance
  - Lifecycle and Dependency management
  - Analysis and reporting on the meta-model
- Non-blocking asynchronous routing
- Distributed architecture

Running your SOA like a Web startup



http://www.zdnet.com/blog/hinchcliffe/running-your-soa-like-a-web-startup/525

API

- **An API** is a business capability delivered over the Internet to internal or external consumers
  - Network accessible function
  - Available using standard web protocols
  - With well-defined interfaces
  - Designed for access by third-parties

What is different from an API and a Service?

- Publishing your API in a Portal
- Expecting people to use it without them having to meet with you
- Making it easy to consume (JSON? Ready built clients in Github?)
- Governance
  - Caring who uses it
  - Letting them know when you version it
  - Meeting an SLA

Key API technologies

- json / rest
- OAuth / OAuth2 keys
- SLA management
- API portal / API Store
  - Catalogue, subscription/purchase
  - Monetization
  - Forum, Ratings, Social
- Analytics


OAUTH
OAUTH

REST description

Swagger: A simple, open sta

swagger.wordnik.com

WSO2, Inc. Mail – Int | Offline WSO2 Mail | WSO2, Inc. – Calend | Fremantle Family Ma | Shorten with bit.ly | Gmail – Inbox (720)

swagger

Home | Specification | Downloads | Demo →

Swagger API Explorer

my-awesome-api.com

user — Show/Hide | List Operations | Expand Operations | Raw

word — Show/Hide | List Operations | Expand Operations | Raw

GET	/word.json/{word}/entries	Return entries for a word
GET	/word.json/{word}/examples	Returns examples for a word
POST	/word.json/{word}/examples	Fetches examples for a word
POST	/word.json/{word}/wordForms	Adds a Relationship Map to a word
GET	/word.json/{word}/wordForms	Returns other forms of a word
DELETE	/word.json/{word}/wordForms	Deletes a relationship from a word
GET	/word.json/{word}	Given a word as a string, returns the WordObject that represents it

Parameters

Parameter	Value	Description
word	(required)	String value of WordObject to return
useCanonical		If true will try to return the correct word root ('cats' -> 'cat'). If false returns exactly what was requested.
includeSuggestions		Return suggestions (for correct spelling, case variants, etc.)
shouldCreate		Create word if not existing

Try it out!

GET	/word.json/{word}/definitions	Return definitions for a word
GET	/word.json/{word}/stats	Returns word statistics

Document your API with Sty

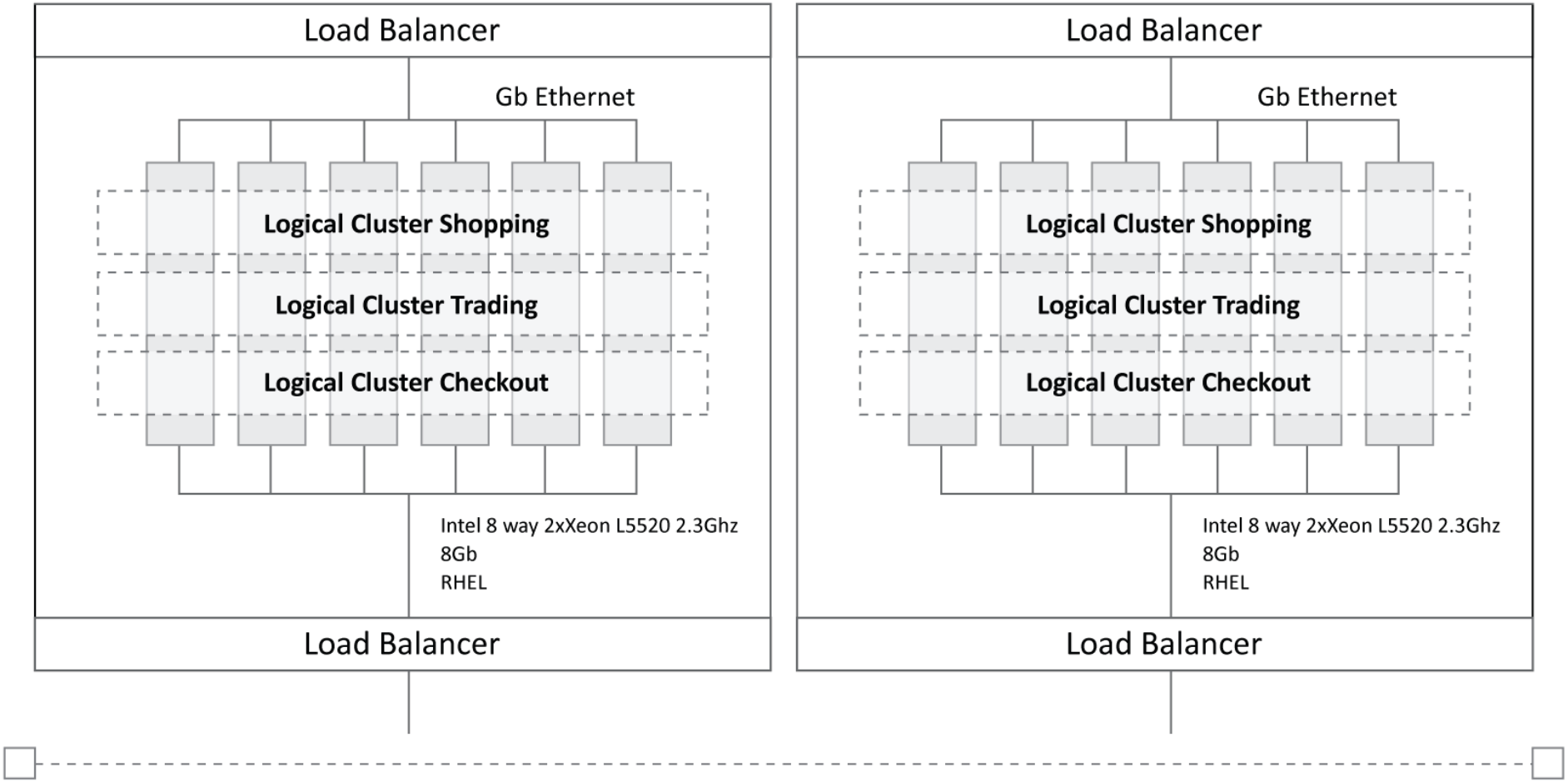
Swagger is a specification and complete framework implementation for describing, producing, consuming, and visualizing RESTf web services. The overarching goal of Swagger to enable client and documentation systems to update at the same pace as the server. The documentation of methods, parameters and models are tightly integrated into the server coc allowing APIs to always stay in sync. With Swagg deploying managing, and using powerful APIs ha never been easier.

Who is responsible for Swagger?

Both the specification and framework implementation are initiatives from Wordnik. Swagger was developed for Wordnik's own use during the development of developer.wordnik.c and the underlying system. Swagger developme began in early 2010—the framework being released is currently used by Wordnik's APIs, wh power both internal and external API clients.

Why is Swagger useful?

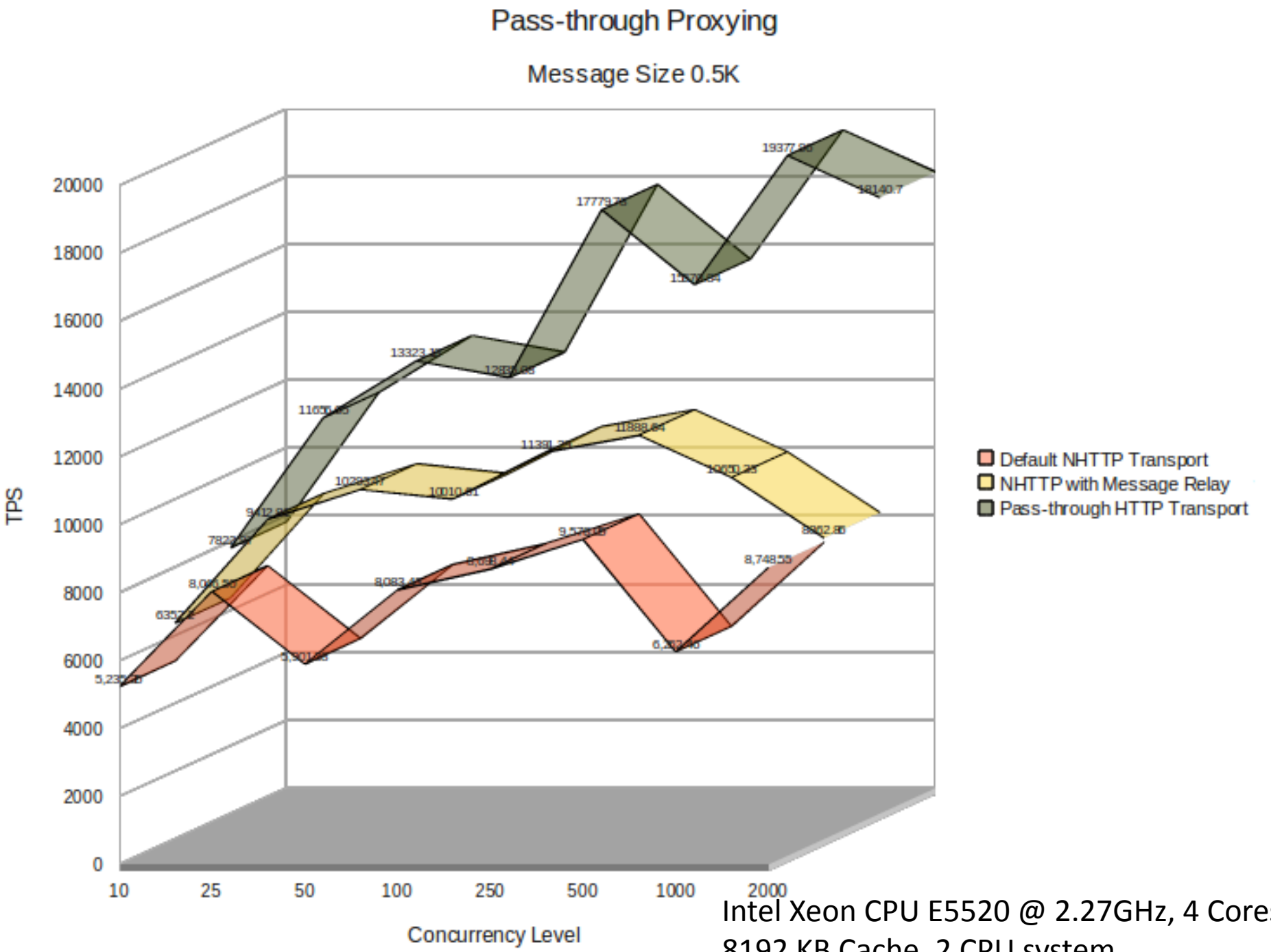
High volume integration @ eBay



100's of service instances
Tomcat + Axis2

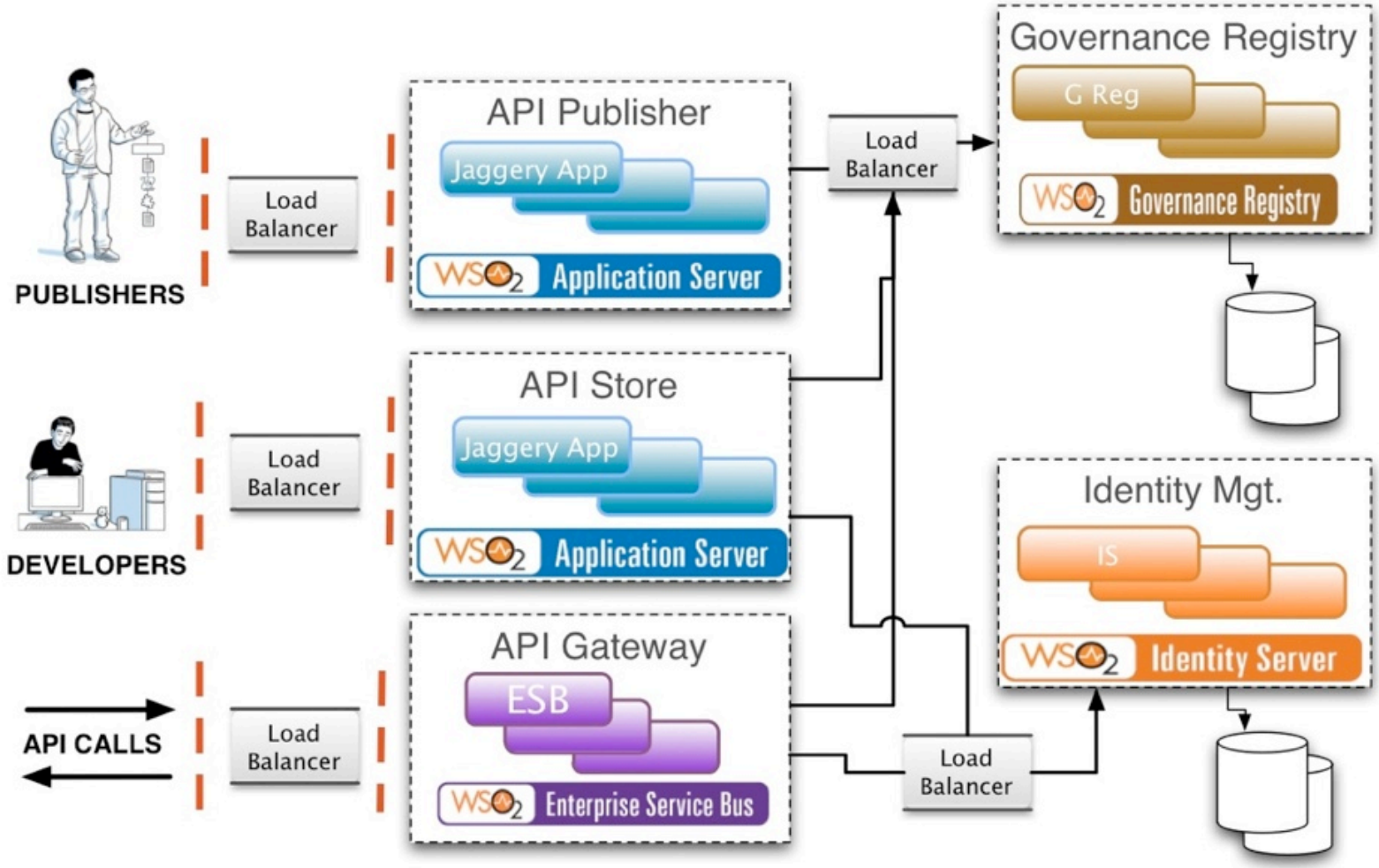
Change in focus

- Security, tokens, access control/entitlement
- Throttling, caching
- Latency and CPU usage
- Monitoring, BAM and CEP

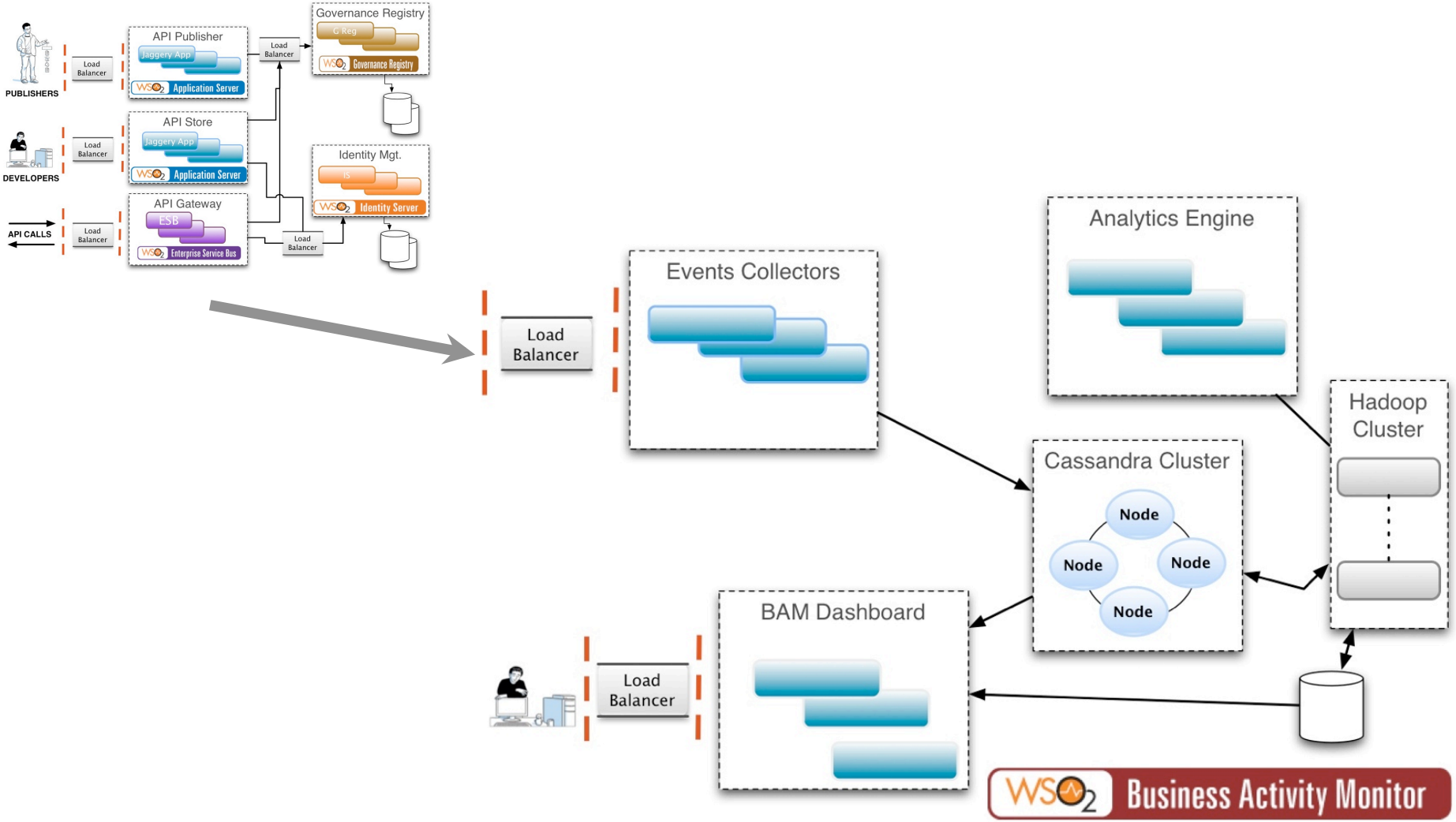


Intel Xeon CPU E5520 @ 2.27GHz, 4 Cores, 8192 KB Cache. 2 CPU system.

API management

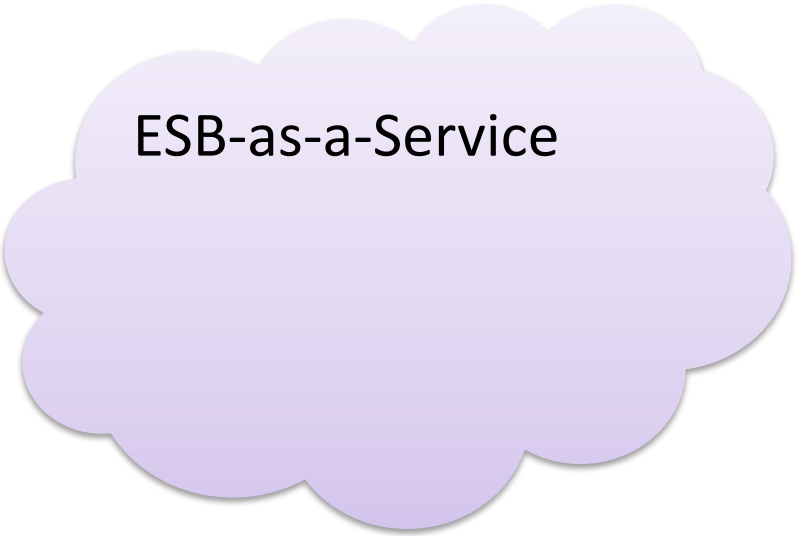


Scalable analytics



Cloud integration

salesforce.com



Cloud Service Gateway

Database

SAP

Cloud integration

- APIs are the right approach
  - Use a “cloud gateway” to bridge into internal systems
- “Push-me pull-you” pattern
  - Use an active ESB in the cloud
- Analytics
  - See what is happening

What's next?

- Still a long way from canonical models
- Successful systems are using "soa" and "rest" at scale
  - Architecture is more important than dogma
- Governance sounds boring but is key
- Applying monetization approaches and "API Store" models
- Analytics and feedback loops

Summary

- Integration has evolved in some interesting ways
  - Async messaging, EDA, APIs, High Volume
- Evolution isn't monotonic
- Doing APIs right is about the mindset as much as the technology