Data Integration and Large-scale Analysis (DIA)
03 Replication and Message-oriented Middleware

Prof. Dr. Matthias Boehm
Technische Universität Berlin
Berlin Institute for the Foundations of Learning and Data
Big Data Engineering (DAMS Lab)

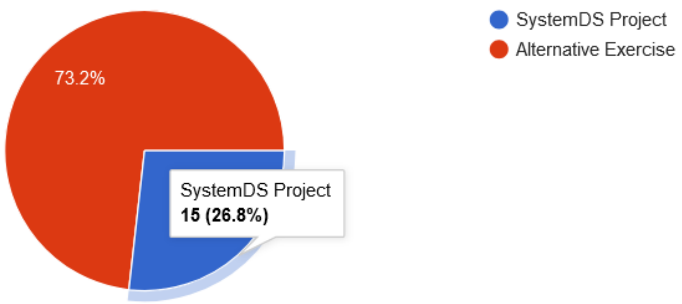
Last update: Oct 31, 2024

Announcements / Administrative Items

- **#1 Video Recording**
  - **Hybrid lectures:** in-person H 0107, zoom live streaming, video recording
  - https://tu-berlin.zoom.us/j/9529634787?pwd=R1ZsN1M3SC9BOU1OcFdmem9zT202UT09

- **#2 Project Selection**
  - Binding project/exercise selection by **Oct 31**
  - Via the following form (**preferred**), or via email to matthias.boehm@tu-berlin.de (in case of problems)

https://forms.gle/w3oREjTTsMYp6stu9

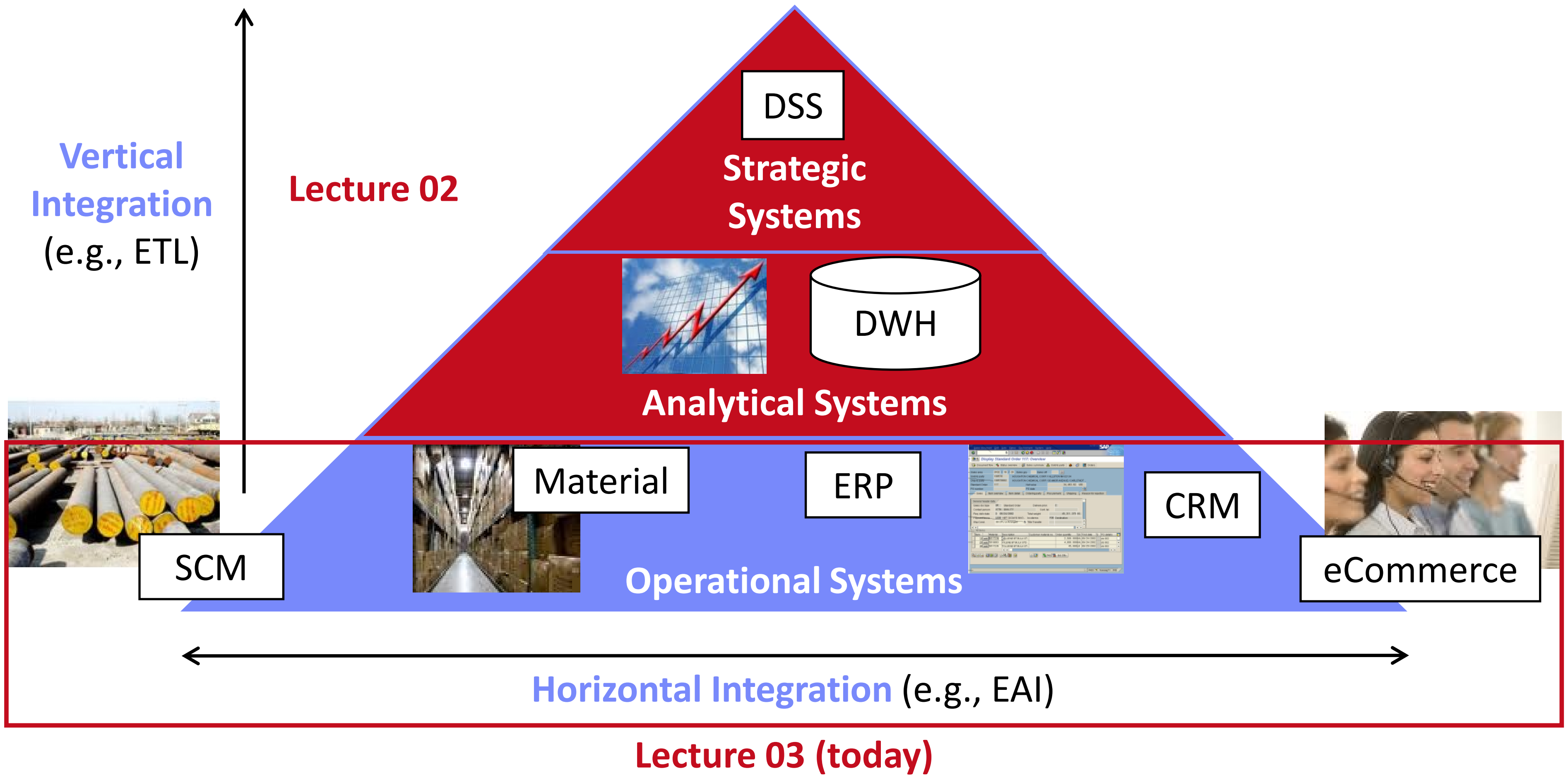


Agenda

- Motivation and Terminology
- Distributed TX & Replication Techniques
- Asynchronous Messaging
- Message-oriented Integration Platforms

Motivation and Terminology

Recap: Information System Pyramid

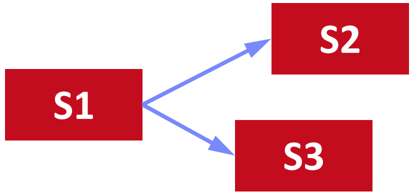


Messaging

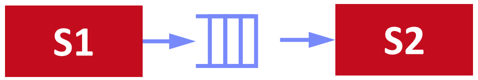
[**Credit:** https://www.gstatic.com/onebox/dictionary/etymology]

- **Def: Message**
  - Piece of information in certain structure
  - Send from source (transmitter) over channel to destination (receiver)
  - **Syntax:** different message formats (binary, text, XML, JSON, Protobuf)
  - **Semantic:** different domain-specific message schemas (aka data models)

- **Synchronous Messaging**
  - **Strict consistency requirements**
  - Overhead for distributed transactions via 2PC
  - Low local autonomy, usually data-driven



- **Asynchronous Messaging**
  - **Loose coupling**, **eventual consistency** requirements
  - Batching for efficient replication and updates
  - Latency of update propagation



Types of Data Formats

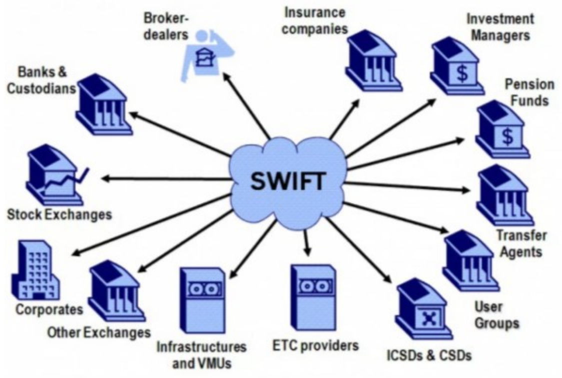
- **General-Purpose Formats**
  - **CLI/API** access to DBs, KV-stores, doc-stores, time series DBs, etc
  - **CSV** (comma separated values)
  - **JSON** (javascript object notation), **XML**, **Protobuf**
- **Sparse Matrix Formats**
  - **Matrix market:** text IJV (row, col, value)
  - **Libsvm:** text compressed sparse rows
  - Scientific formats: **NetCDF**, **HDF5**

```
%%MatrixMarket matrix coordinate real general
% -------------------------------------------
% 0 or more comment lines
% -------------------------------------------
5  5  8
1 1 1.000e+00
2 2 1.050e+01
3 3 1.500e-02
1 4 6.000e+00
4 2 2.505e+02
4 4 -2.800e+02
4 5 3.332e+01
5 5 1.200e+01
```

- **Large-Scale Data Formats**
  - ORC, **Parquet** (column-oriented file formats)
  - **Arrow** (cross-platform columnar in-memory data)
- **Domain-specific Formats: often binary, structured text, XML**

Example Domain-specific Message Formats

- **Finance: SWIFT**
  - Society for Worldwide Interbank Financial Telecommunication
  - >10,000 orgs (banks, stock exchanges, brokers and traders)
  - Network and message formats for financial messaging
  - MT and MX (XML, ISO 20022) messages



[https://ihodl.com]

- **Health Care: HL/7, DICOM**
  - Health Level 7 (HL7) messages for clinical/admin data exchange (v2.x structured text msgs, v3 XML-based msgs)
  - Digital Imaging and Communications in Medicine (DICOM)

- **Automotive: ATF, MDF**
  - Association for Standardisation of Automation and Measuring Systems (ASAM)
  - E.g., Open Transport Data Format (ATF), Measurement Data Format (MDF), calibrations (CDF), auto-lead XML (ADF), open platform communications (OPC)

➔ **Sometimes Large-scale analytics over histories of messages** (e.g., health care analytics, fraud detection, money laundering)

Types of Message-Oriented Middleware

- **#1 Distributed TXs & Replication**

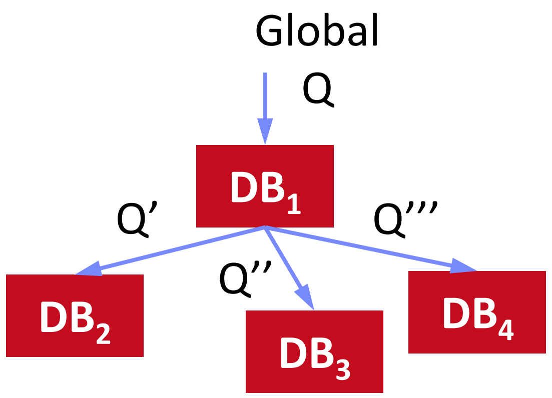
- **#2 Message Queueing**
  - Persistent message queues with well-defined delivery semantics
  - Loose coupling of connected systems or services (e.g., availability)

- **#3 Publish Subscribe**
  - Large number of subscribers to messages of certain topics/predicates
  - Published messages forwarded to qualifying subscriptions

- **#4 Integration Platforms**
  - Inbound/outbound adapters for external systems
  - Sync and async messaging, message transformations, enrichment

Distributed TX & Replication Techniques

Distributed Database Systems



- **Distributed DBS**
  - Distributed database: Virtual (logical) database that appears like a local database but consists of multiple physical databases
  - Multiple local DBMS, components for global query processing
  - **Terminology:** **virtual DBS** (homogeneous), **federated DBS** (heterogeneous)

- **Challenges**
  - **Tradeoffs:** Transparency – autonomy, **consistency – efficiency/fault tolerance**
  - **#1** Global view and query language → schema architecture
  - **#2** Distribution transparency → global catalog
  - **#3** Distribution of data → data partitioning
  - **#4** Global queries → distributed join operators, etc
  - **#5** Concurrent transactions **→ 2PC**
  - **#6** Consistency of copies **→ replication**

Beware: Meaning of "Transparency" (invisibility) here

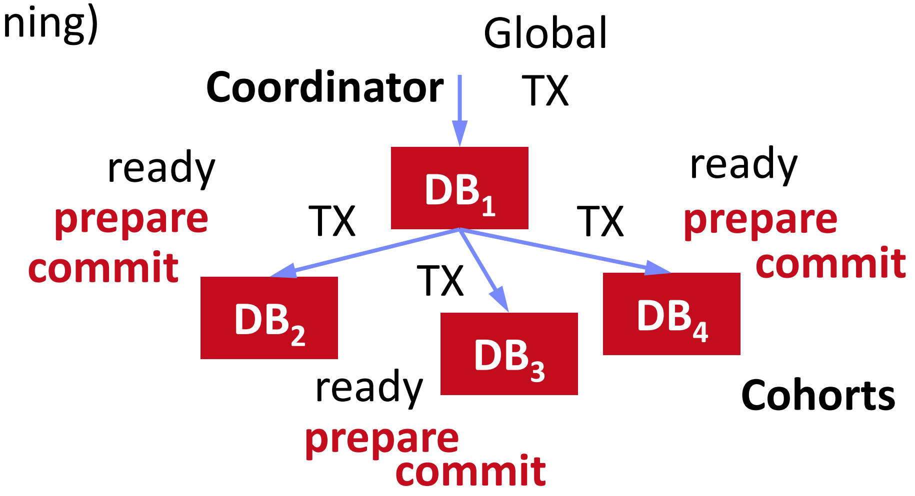
Two-Phase Commit (2PC)

- **Recap: Database Transaction**
  - A transaction (TX) is a **series of steps** that brings a database from a **consistent state** into another (not necessarily different) **consistent state**
  - **ACID properties** (atomicity, consistency, isolation, durability)

- **Problems in Distributed DBS**
  - Node failures, and communication failures (e.g., network partitioning)

  ➔ **Distributed TX processing to ensure consistent view** (atomicity/durability)

- **Two-Phase Commit** (via 4*(n-1) msgs)
  - **Phase 1 PREPARE:** check for successful completion, logging
  - **Phase 2 COMMIT:** commit/abort, release locks, and other cleanups
  - What happens if nodes unavailable, or report errors on prepare



Two-Phase Commit (2PC), cont.

- **Excursus: Wedding Analogy**
  - Coordinator: marriage registrar
  - **Phase 1:** Ask for willingness
  - **Phase 2:** If all willing, declare marriage
- **#1 Problem: Many Messages**
  - 4(n-1) messages in successful case, otherwise additional msgs
- **#2 Problem: Blocking Protocol**
  - Local node PREPARE → FAILED ➔ TX is guaranteed to be aborted
  - Local node PREPARE → READY ➔ waiting for global response
  - Failure of coordinator+cohort, or participating coordinator → **outcome unknown**
- **Other Problems**
  - Atomicity in heterogeneous systems w/o XA
  - Deadlock detection, optimistic concurrency control, etc

Note: APIs for automatic vs programmatic 2PC

Extended Distributed Commit Protocols

- **2PC Improvements**
  - **Hierarchical Commit:** establish message tree from coordinator to local nodes
    ➔ parallelization of message handling over inner nodes
  - **Presumed Abort:** assume abort if there are no commit log entries
    ➔ asynchronous logging of aborts, no ACK on abort

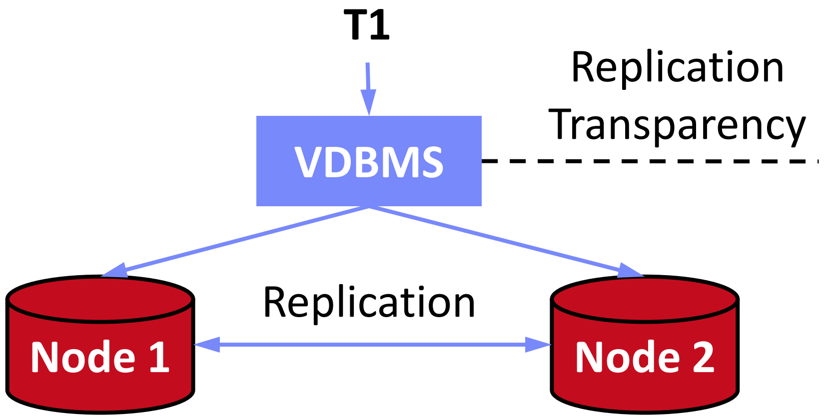
- **1PC (fewer messages)**
  - Combine TX operations w/ PREPARE to reduce 2(n-1) messages
  - Local nodes enter waiting state earlier

- **3PC (non-blocking)**
  - a) CAN COMMIT? Yes/no
  - b) PREPARE COMMIT? Ack
  - c) COMMIT? Ack
  - Cohorts can collectively decide on commit if at least one in PREPARE-COMMIT

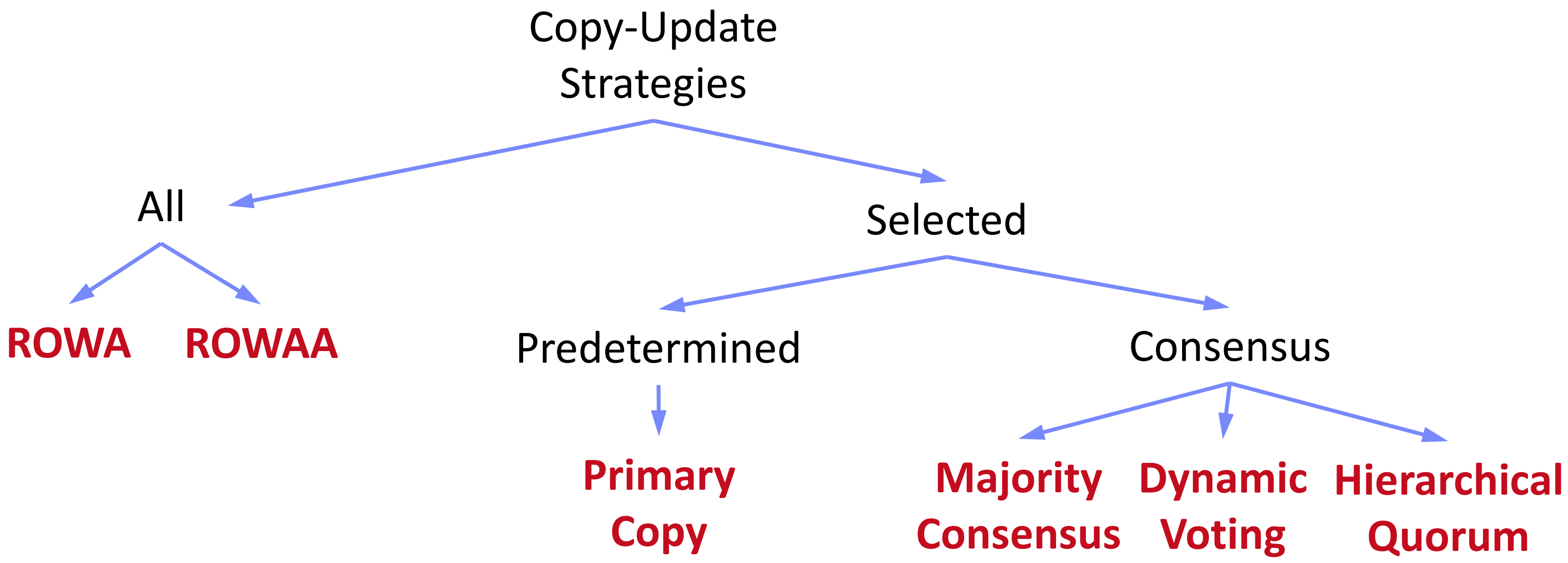
Protocol	# Msgs
1PC	2(n-1)
2PC	4(n-1)
3PC	6(n-1)

Replication Overview

- **Replication**
  - Redundancy of stored fragments
  - Availability/efficiency (read) vs update overhead / storage

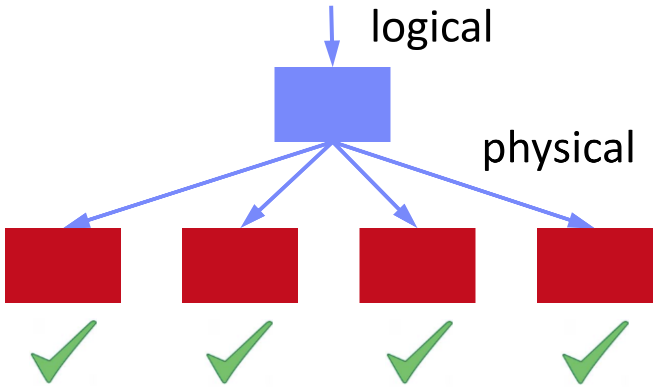


- **Replication Techniques**

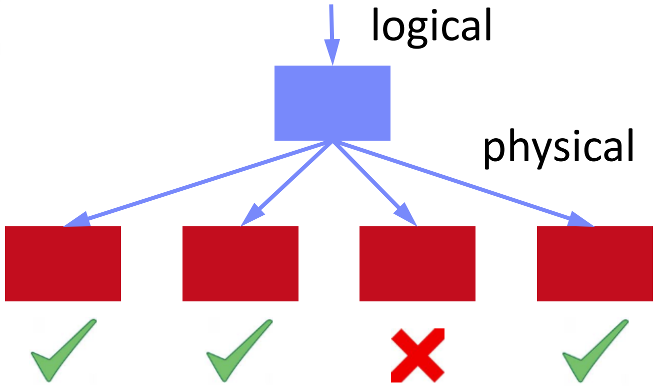


Replication Techniques

- **ROWA**
  - Read-One/Write-All
  - **Read:** good performance/availability
  - **Write:** high overhead and only successful if all available



- **ROWAA**
  - Read-One/Write-All-Available
  - **Relaxed availability requirement** for write operations

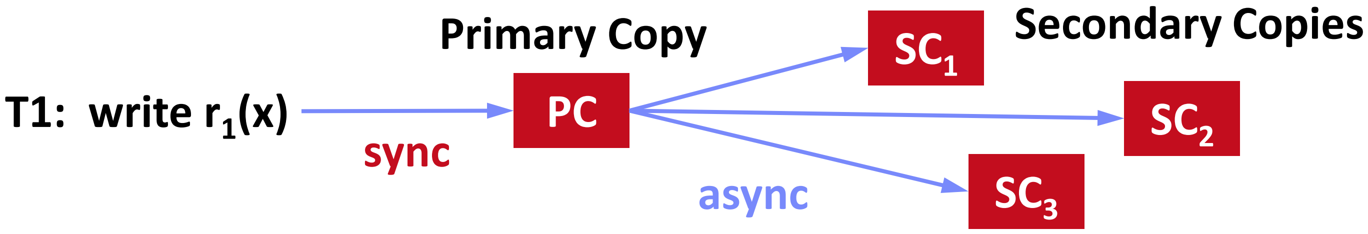


„Update anywhere-anytime-anyway transactional replication has unstable behavior as the workload scales up: **a ten-fold increase in nodes and traffic gives a thousand fold increase in deadlocks or reconciliations**. Master copy replication (**primary copy**) schemes reduce this problem."

[**Jim Gray**, Pat Helland, Patrick E. O'Neil, Dennis Shasha: The Dangers of Replication and a Solution, **SIGMOD 1996**]

Replication Techniques, cont.

- **Primary Copy**
  - Update single primary copy **synchronously**
  - **Asynchronous propagation** of updates to other replicates, read from all



  - **Pro:** Higher update performance, good locality, and availability
  - **Con:** Potentially stale read on secondary copies (w/ and w/o locks)
  - **Load balancing:** place PC of different objects on different nodes

Replication Techniques, cont.

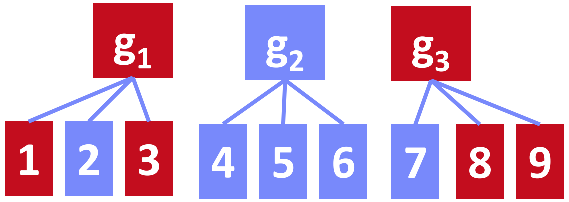
- **Consensus Protocols**
  - **Basic idea:** voting if read/write access is permissible (w.r.t. serializability)
  - Each replicate has vote → all votes Q
  - Read quorum Q_R and write quorum Q_W

Overlap Rules:

$Q_R + Q_W > Q$

$Q_W > Q/2$

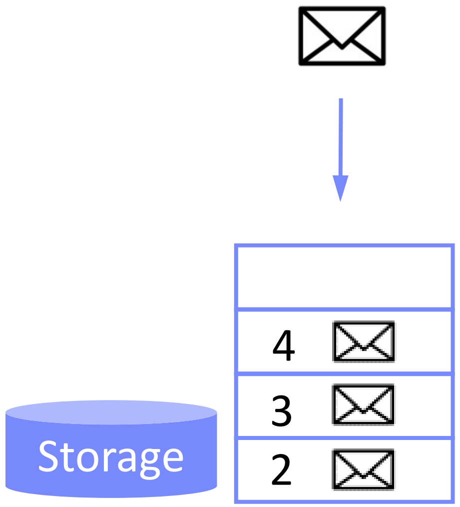
- **#1 Majority Consensus**
  - Read requires **$Q_R > Q/2$**, lock all and read newest replica
  - Write requires **$Q_W > Q/2$**, lock and update all
- **#2 Dynamic Quorums**
  - Problem: network partitioning → retain vote for updated replica
- **#3 Hierarchical Quorums**
  - Obtain majority of nodes (here **two**) in multiple levels of the tree



Asynchronous Messaging

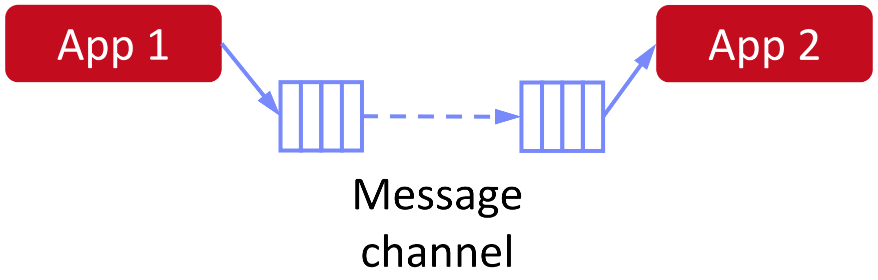
Message Queueing

- **Message**
  - Atomic packet of data + meta data, wrapped as a message



- **Message Queue**
  - FIFO or priority queue of messages
  - In-memory, sometimes with persistent storage backend and transactional semantics
  - Internal IDs, receive time

- **Remote Message Queues**
  - Loose coupling of applications (no direct API calls, etc)
  - Independent of HW and OS



Recap: Message Delivery Guarantees

- **#1 At Most Once**
  - **"Send and forget"**, ensure data is never counted twice
  - Might cause data loss on failures

- **#2 At Least Once**
  - **"Store and forward"** or acknowledgements from receiver, replay stream from a checkpoint on failures
  - Might create incorrect state (processed multiple times)

- **#3 Exactly Once**
  - **"Store and forward" w/ guarantees** regarding state updates and sent msgs
  - Often via dedicated transaction mechanisms

BREAK and Test Yourself!

- Assume a message-oriented middleware with a single FIFO message queue. Indicate, in the table below, true (✓) properties of the following three message delivery guarantees. [5 points]

	At Most Once	At Least Once	Exactly Once
Requires Message Persistence		✓	✓
Requires Delivery TX Mechanism			✓
Prevents Message Outrun	✓	✓	✓
Prevents Message Loss		✓	✓
Prevents Message Double Delivery	✓		✓

Example Systems

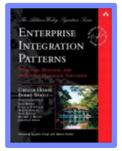
- **IBM MQSeries**
  - Message-oriented middleware for async queue communication
  - Connections/objects: **MQCONN**, MQDISC, MQOPEN, MQCLOSE
  - Queue ops: MQCRTMH, **MQPUT**, **MQGET**, MQSET, MQINQ, MQSTAT
  - Transactions: MQBEGIN, MQBACK, MQCMIT

- **JMS (Java Message Service)**
  - J2EE API of messaging services in Java (messages, queues, sessions, etc)
  - JMS providers: e.g., **IBM Websphere MQ**, **Apache ActiveMQ, RabbitMQ**

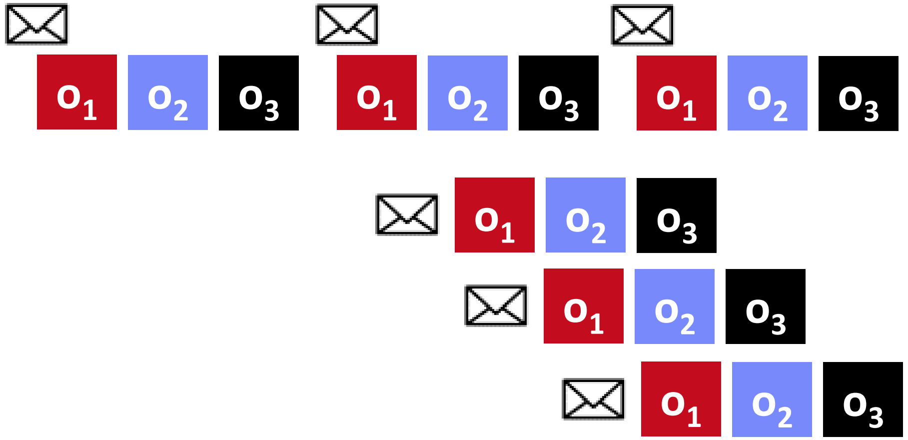
- **AWS Simple Queueing Service (SQS)**
  - Message queueing service for loose coupling of micro services
  - Default queue: best effort order, **at-least-once**, high throughput
  - FIFO: guarantees FIFO order, and **exactly-once**

Parallel Message Processing

[Gregor Hohpe, Bobby Woolf: Enterprise Integration Patterns, Addison-Wesley, 2004]

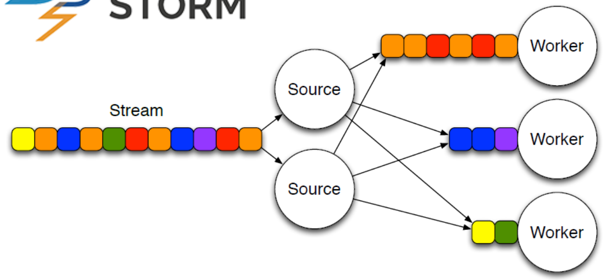


- **#1 Pipeline Parallelism**
  - **"Pipes and filters"**: leverage pipeline parallelism of chains of operators
  - More complex w/ routing / control flow (possible via punctuations)



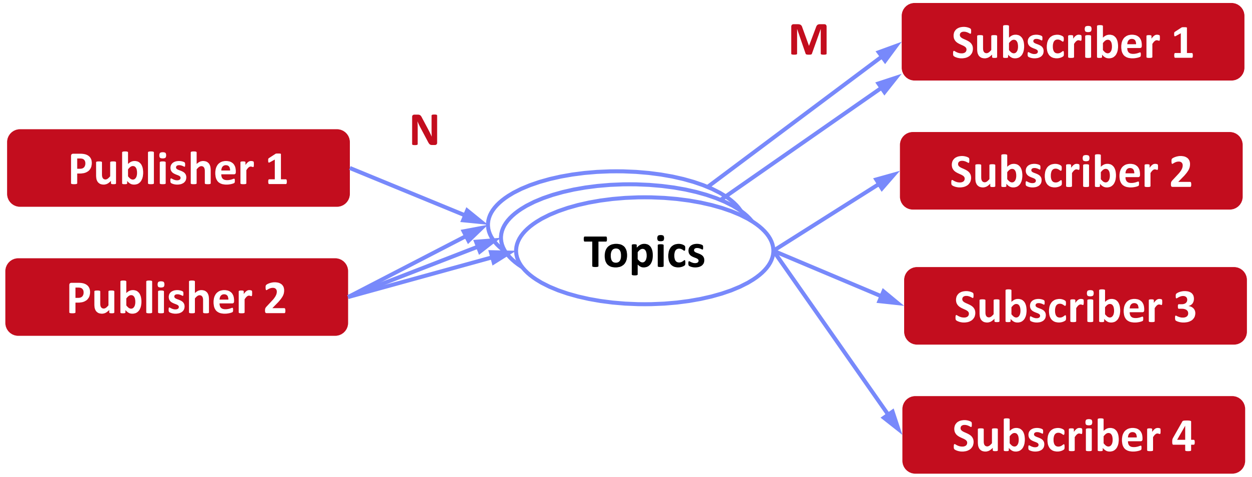
- **#2 Operator Parallelism**
  - Multi-threaded execution of multiple messages within one operator (pattern **"competing consumers"**)
  - Requires robustness against partial out-of-order, or resequencing

- **#3 Key Range Partitioning**
  - Explicit routing to independent pipelines (patterns **"message router"**, **"content-based router"**)
  - Ordering requirements only within each pipeline



Publish/Subscribe Architecture

- **Overview Publish-Subscribe (Pub/Sub)**



- **Key Characteristics**
  - Often imbalance between few publishers and many subscribers
  - Topics: explicit or implicit (e.g., predicates) groups of messages to publish into or subscribe from
  - Addition and deletion of subscribers rare compared to message load
  - ECA (event condition action) evaluation model
  - Often **at-least-once** guarantee

Alternative Exercise:
Streaming Full Text Search
[https://mboehm7.github.io/teaching/ws2425_dia/DIA_2024_Exercise.pdf]

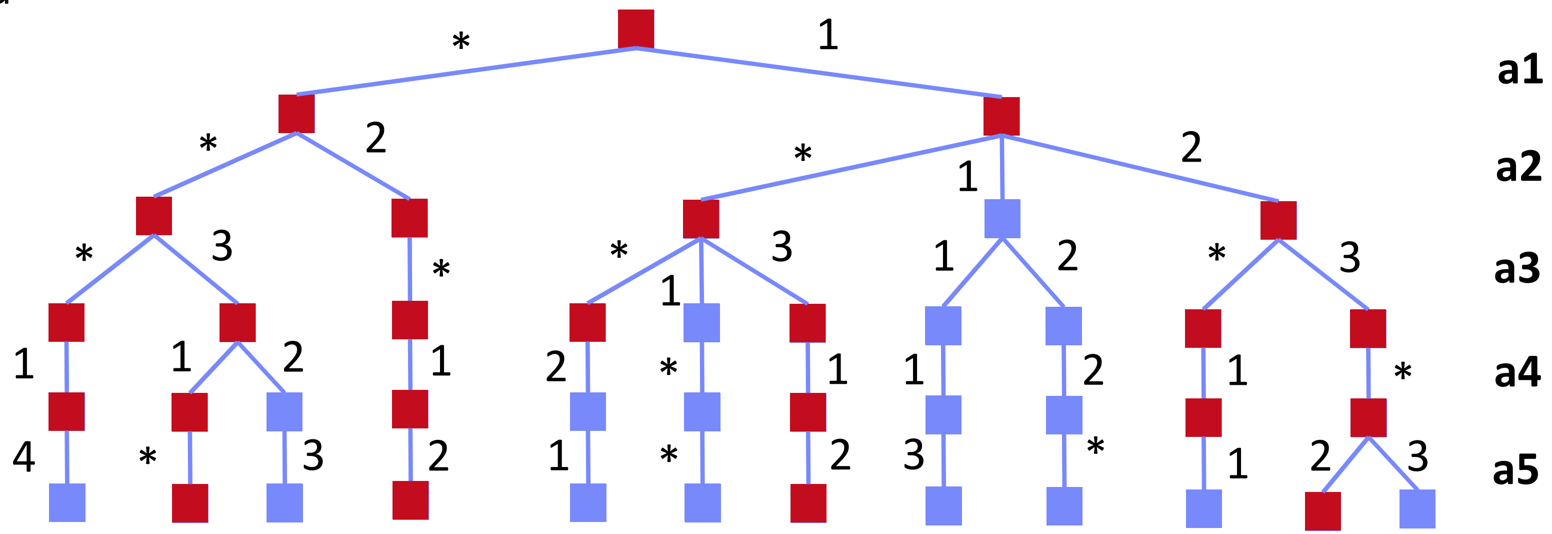
Publish/Subscribe Architecture, cont.

- **Subscriber Filtering**
  - Complex predicates of range filters, equi-predicates, and negation
  - **Goal:** Avoid naïve scan over all subscriber predicates / topics

- **Matching Algorithm**
  - Matching event against a set of subscriptions
  - **Approach:** sorting and parallel search tree

[Guruduth Banavar et al: An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. **ICDCS 1999**]

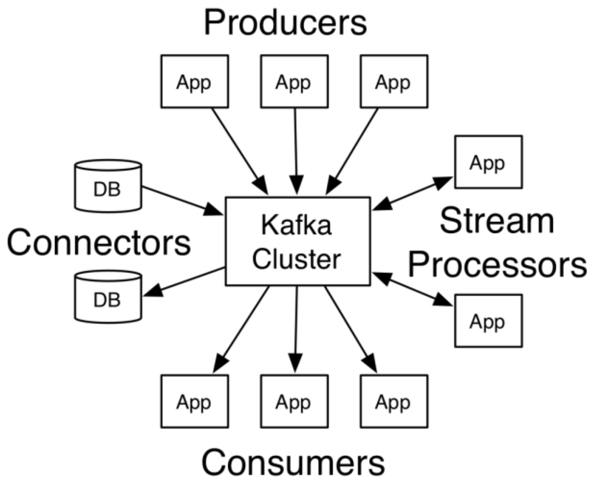
Example Publish
$\{a_1=1, a_2=2, a_3=3, a_4=1, a_5=2\}$



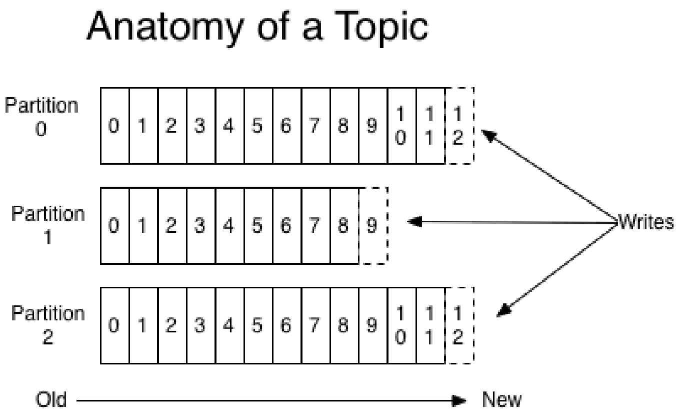
Apache Kafka

[https://kafka.apache.org/documentation]

- **Overview System Architecture**
  - **Publish & Subscribe** system w/ partitioned topics
  - **Storage of data streams** in distributed, fault-tolerant cluster (replicated)
  - Configurable **retention periods** (e.g., days)
  - **APIs:** producer API, consumer API, streams API, Connector API



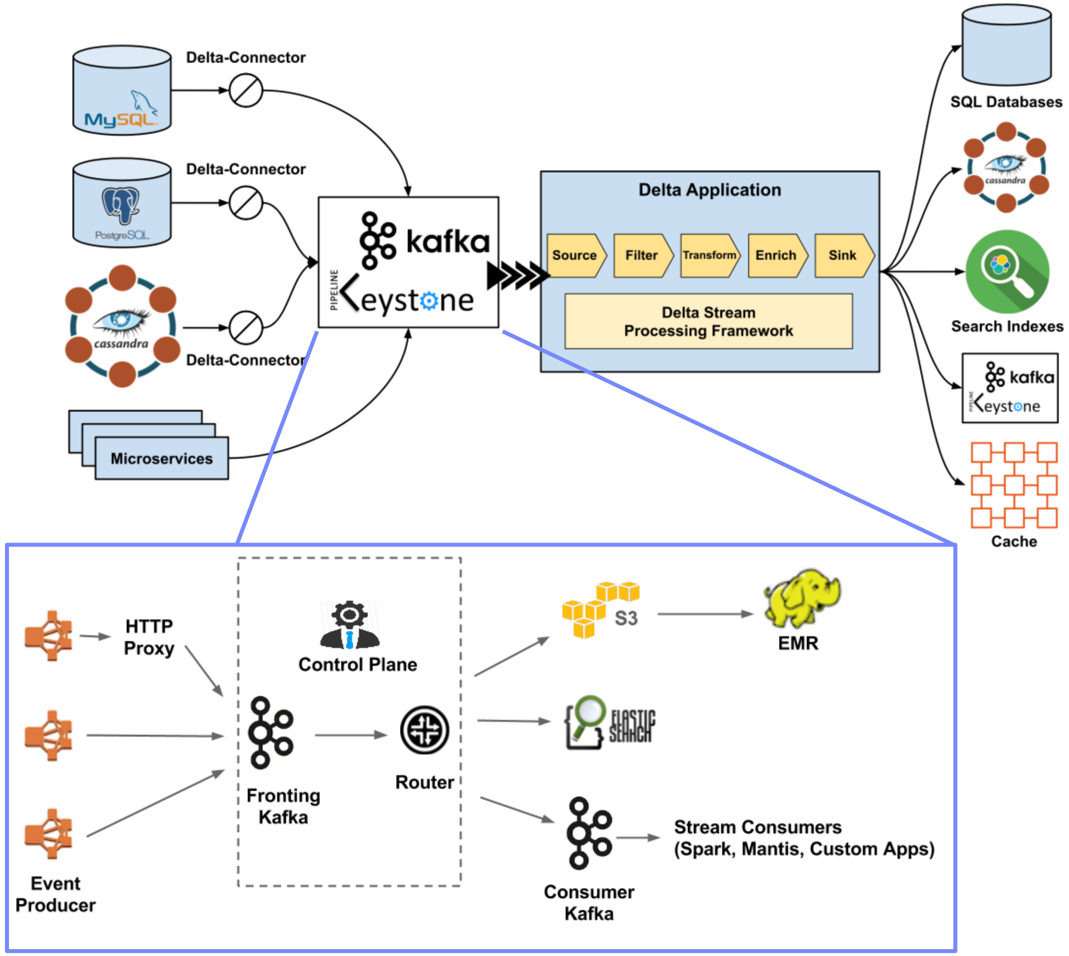
- **Topics**
  - Explicit categories w/ user-defined (semantic) partitioning
  - Partitions are ordered, immutable sequences of records (log) w/ **offsets**
  - Current **offset** per consumer stored



Apache Kafka, cont.

[https://medium.com/netflix-techblog/delta-a-data-synchronization-and-enrichment-platform-e82c36a79aee, Oct 15 2019]

- **Netflix Delta**
  - **A Data Synchronization and Enrichment Platform**
  - DSL and UDF APIs for custom filters and transformations



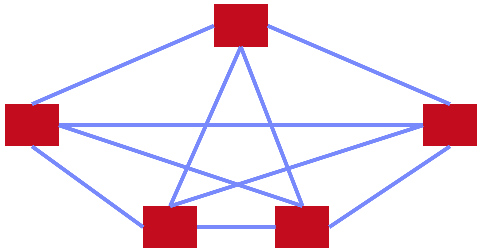
- **Netflix Keystone** (Kafka frontend)
  - **~500G events/day** (5M events/s peak)
  - **~1.3PB/day**

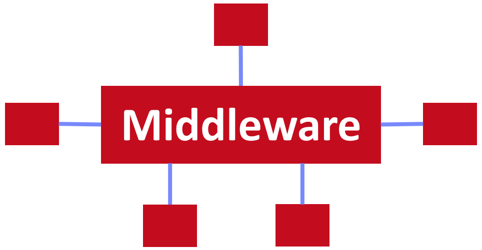
[https://medium.com/netflix-techblog/evolution-of-the-netflix-data-pipeline-da246ca36905]

Message-oriented Integration Platforms

Overview Message-oriented Integration Platforms

- **Motivation**
  - Integration of many applications and systems via common IR
  - **Beware:** syntactic vs semantic data models

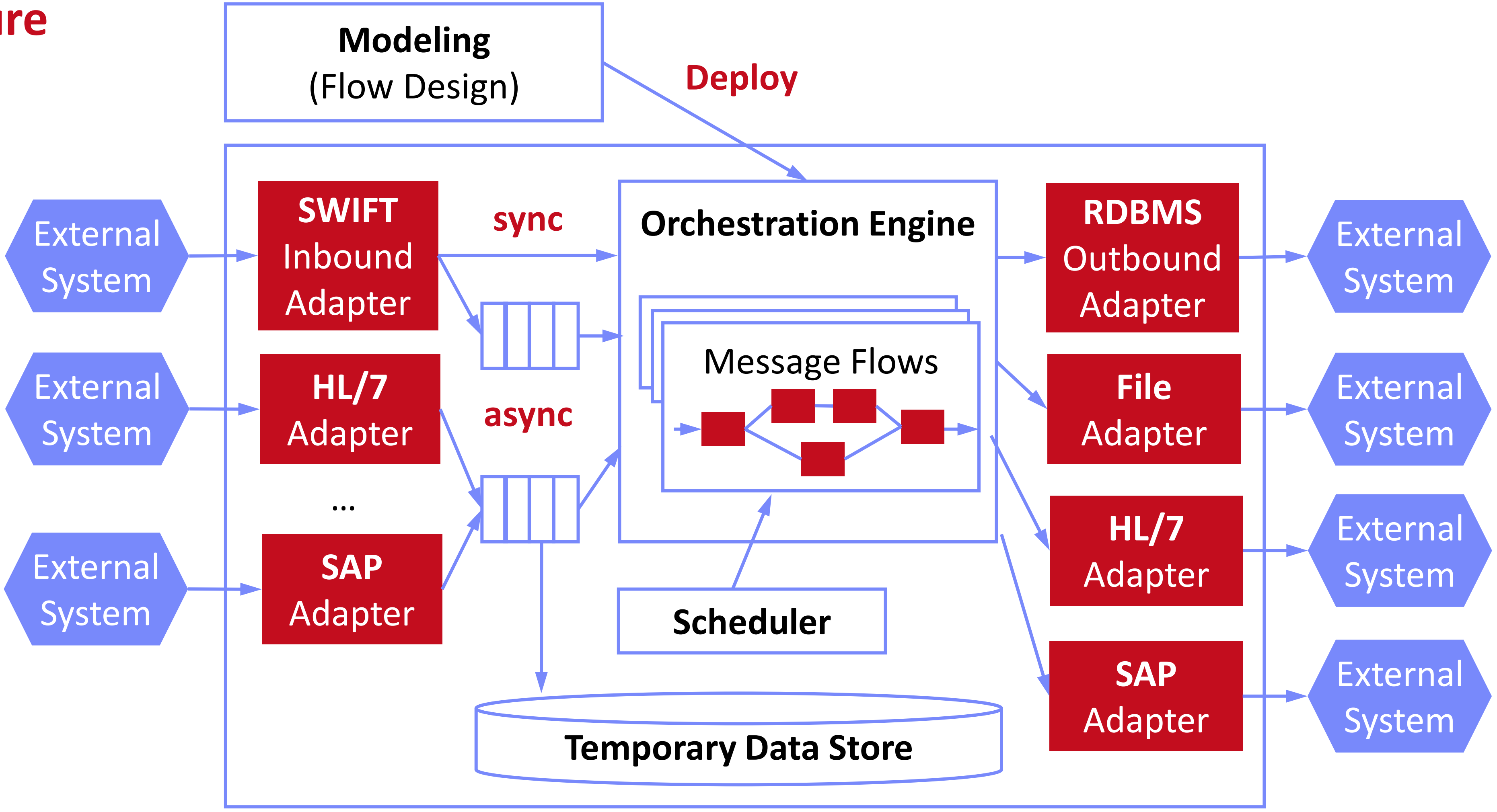




- **Evolving Names**
  - **Enterprise Application Integration** (EAI)
  - **Enterprise Service Bus** (ESB)
  - **Message Broker**

- **Example Systems**
  - IBM App Connect Enterprise (aka Integration Bus, aka Message Broker)
  - MS Azure Integration Services + Service Bus (aka Biztalk Server)
  - SAP Process Integration (aka Exchange Infrastructure)
  - SQL AG TransConnect

Common System Architecture

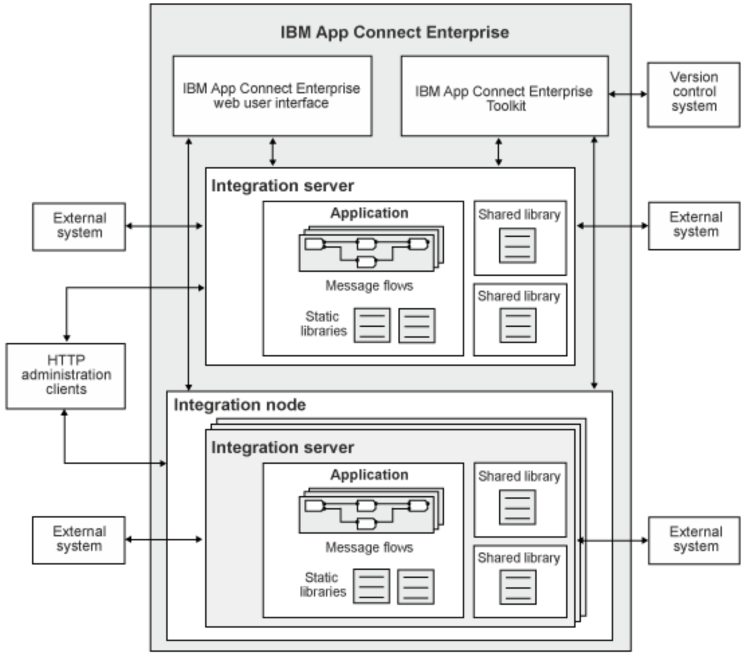


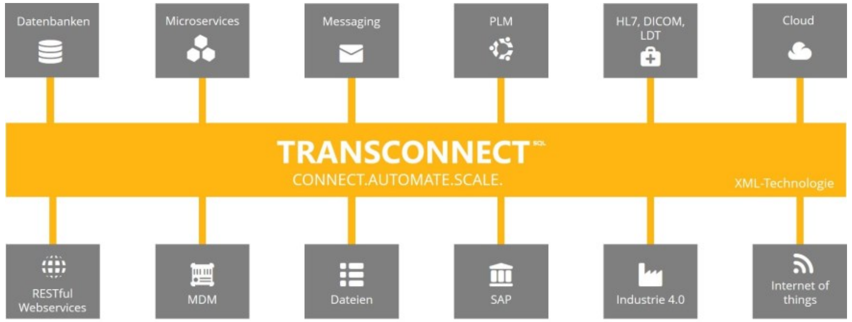
Common System Architecture, cont.

- **#1 Synchronous Message Processing**
  - **Event: client input message**
  - Client system blocks until message flow executed to output messages delivered to target systems
- **#2 Asynchronous Message Processing**
  - **Event: client input message from queue**
  - Client system blocks until input message stored in queue
  - Asynchronous message flow processing and output message delivery (**streaming**)
  - Optional acknowledgement, when input message successfully processed
- **#3 Scheduled Processing**
  - **Event: time-based scheduled** message flows (CronJobs)
  - Periodic data replication and loading (e.g., **ETL use cases**)

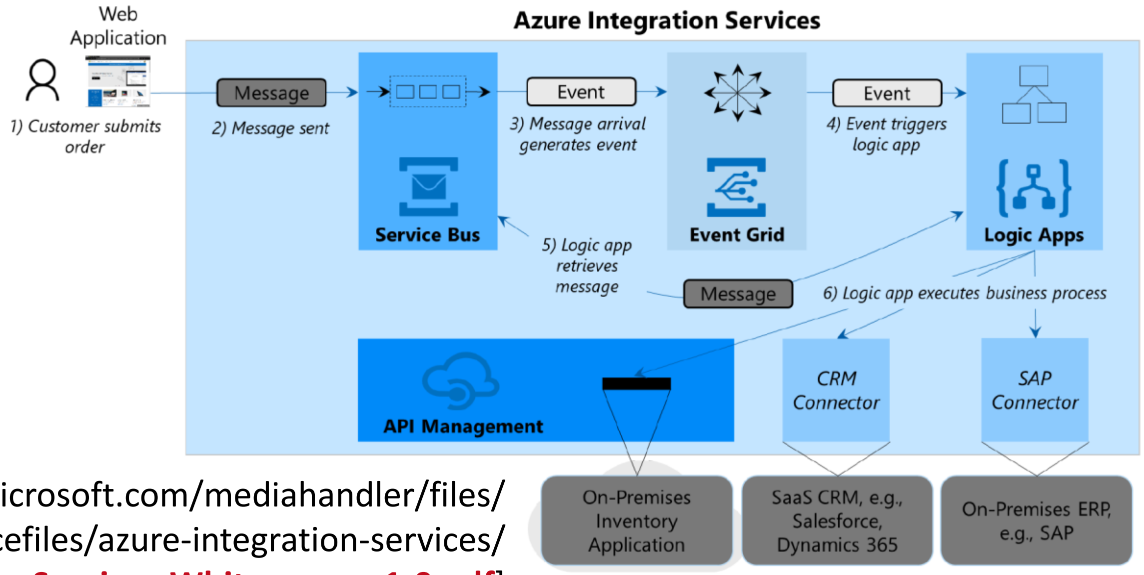
Commercial Systems

[**IBM App Connect Enterprise:** https://www.ibm.com/support/knowledgecenter/en/SSTTDS_11.0.0/com.ibm.etools.mft.doc/ab20551_.htm]





[**SQL AG:** https://www.transconnect-online.de/]

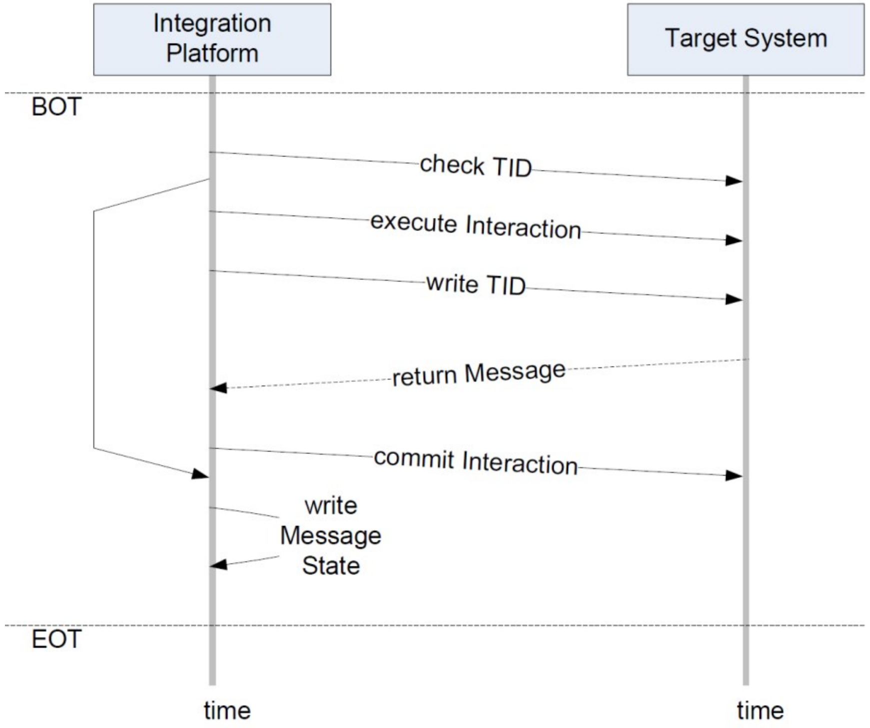


[https://azure.microsoft.com/mediahandler/files/resourcefiles/azure-integration-services/**Azure-Integration-Services-Whitepaper-v1-0.pdf**]

Message Delivery Guarantees, cont.

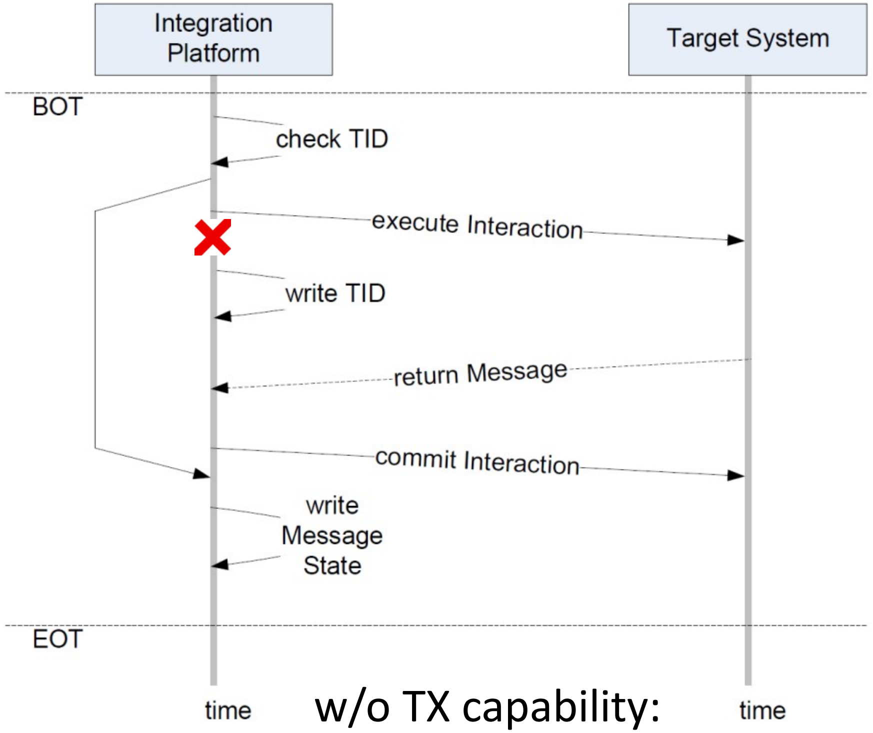
- **Example**
  Exactly-Once

Remote ID Maintenance w/ TX capability



[**Credit:** SQL AG - https://www.transconnect-online.de/]

Local ID Maintenance w/ TX capability



w/o TX capability:
at-least-once

Recap: XML (Extensible Markup Language)

- **XML Data Model**
  - Meta language to define specific **exchange formats**
  - Document format for **semi-structured data**
  - Well formedness
  - XML schema / DTD
- **XPath** (XML Path Language)
  - Query language for **accessing collections of nodes** of an XML document
  - Axis specifies for ancestors, descendants, siblings, etc
- **XSLT** (XML Stylesheet Language Transformations)
  - Schema mapping (transformation) language for XML documents
- **XQuery**
  - Query language to extract, transform, and analyze XML documents

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <student id="1">
    <course id="INF.01014UF" name="Databases"/>
    <course id="706.550" name="AMLS"/>
  </student>
  <student id="5">
    <course id="706.004" name="Databases 1"/>
  </student>
</data>
```

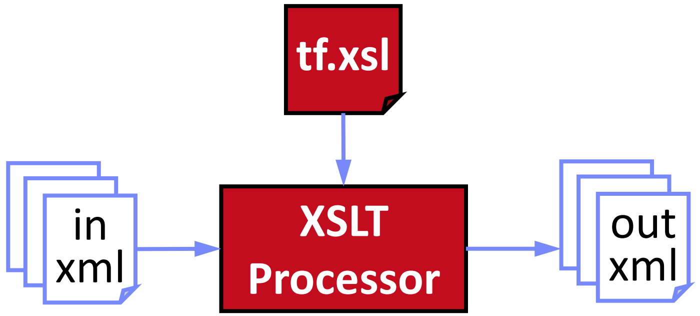
```
/data/student[@id='1']/course/@name
```

→ "Databases"
"AMLS"

XSLT in Integration Platforms

- **Problem**
  - XML often used as **external and internal data representation**
  - Different schemas (message types) ➔ **requires mapping**

- **XSLT Overview**
  - XSLT processor transforms input XML document according to XML stylesheet to output XML documents
  - Subtree specifications via XPath, loops, branches, built-in functions for text processing, etc
  - **Streaming:** STX or XSLT 3.0 streaming
  - **CSV** and **JSON** input/output possible



- **Note: Similar tools/libraries for JSON**

XSLT Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0“ xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <xsl:element name=“suppliers">
    <xsl:for-each select="/resultsets/resultset[@Tablename='Supplier']/row">
      <xsl:element name=“supplier">
        <xsl:attribute name=“ID"><xsl:value-of select="Suppkey"/></xsl:attribute>
        <xsl:element name="Name"><xsl:value-of select="SuppName"/></xsl:element>
        <xsl:element name="Address"><xsl:value-of select="SuppAddress"/></xsl:element>
      </xsl:element>
    </xsl:for-each>
  </xsl:element>
</xsl:template>
</xsl:stylesheet>
```

```
<resultssets>
  <resultset Tablename=“Supplier”>
    <row>
      <Suppkey>7</Suppkey>
      <SuppName>MB</Suppname>
      <SuppAddress>1035 Coleman Rd</SuppAddress>
    </row>
    <row> … </row>
  </resultset>
</resultsets>
```

```
<suppliers>
  <supplier ID=“7”>
    <Name>MB</Name>
    <Address>1035 Coleman Rd</Address>
  </supplier>
  <supplier> … </supplier>
<suppliers>
```

Summary and Q&A

- **Distributed TX & Replication Techniques**
  - Distributed commit protocols
  - Different replication techniques
- **Message-oriented Middleware**
  - Asynchronous Messaging
    (message queueing, publish/subscribe)
  - Message-oriented Integration Platforms
    (system architecture, systems, transformations)

Macroscopic View

- **Next Lectures (Data Integration Architectures)**
  - **04 Schema Matching and Mapping** [Nov 07]
  - **05 Entity Linking and Deduplication** [Nov 14]
  - **06 Data Cleaning and Data Fusion** [Nov 21]
  - **07 Data Provenance and Catalogs** [Nov 28]

Microscopic View