RTI Event Processing

Powered by Coral8

Supreet Oberoi
supreet@rti.com
Vice President, Engineering
Real-Time Innovations, Inc.

Coral8 Quick Overview

- Company
  - Founded in 2003; First GA software release: 2H 2005
  - CEO: Terry Cunningham (Crystal Decisions, Seagate Software, Veritas)
  - 20+ direct customers, 12+ OEMs, 15+ partners
  - 1,800+ active, registered developers
- Market
  - Complex Event Processing software
  - Horizontal customer-base
  - Solution channel focus
- Product
  - Coral8 Release 5.0 (the 9th commercial release)
  - The first commercial declarative SQL-based CEP language
  - High performance CEP Engine that is easy to develop, deploy & scale

RTI

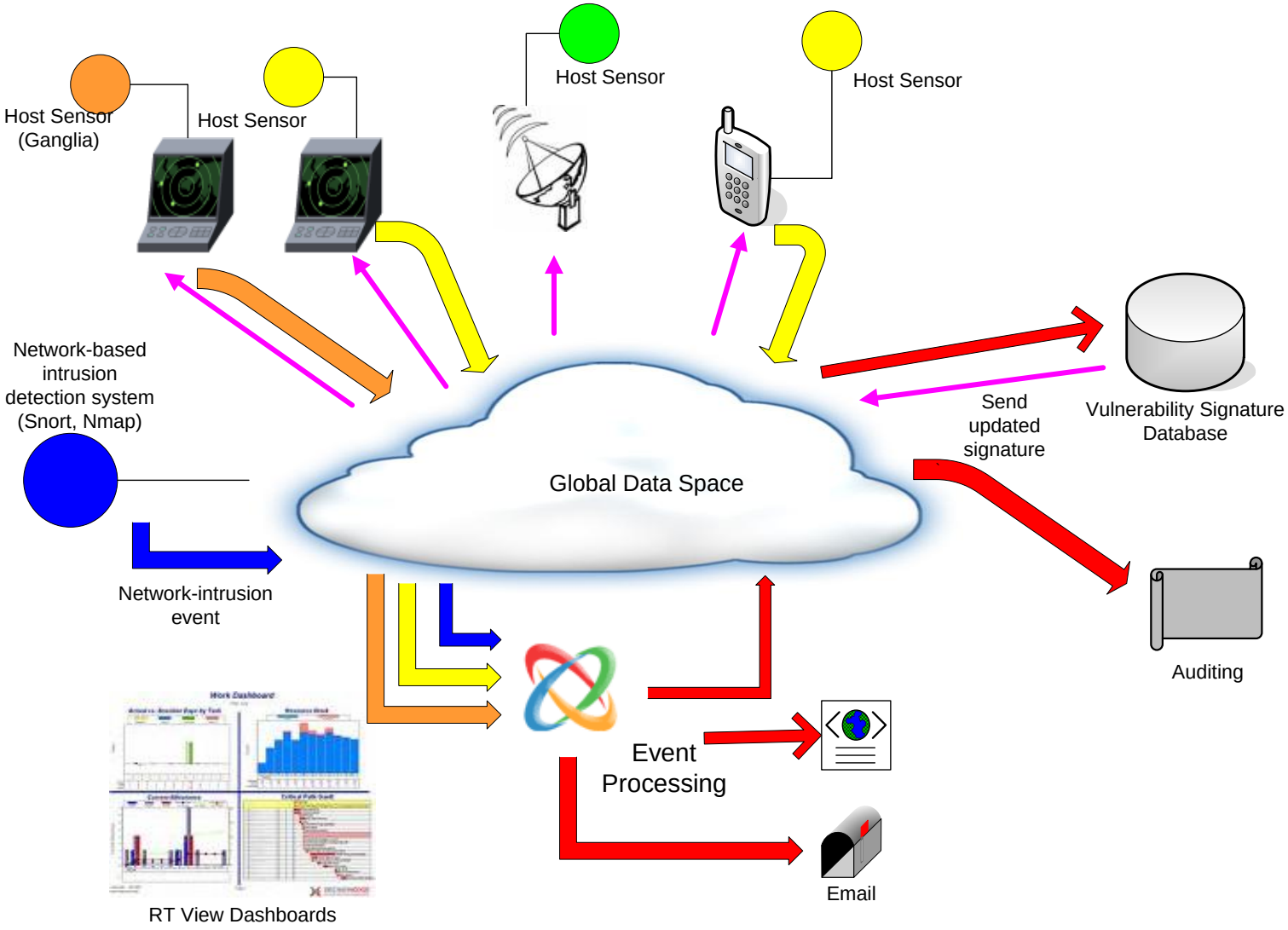
About RTI

- Experts in complex, high performance middleware
- Spun out of Stanford Aerospace Robotics Lab in 1991
- Active contributor to standards
  - Object Management Group (OMG)
  - Network Centric Operations Industry Consortium (NCOIC)
  - Open Group
- Solid financials
  - 15 year track record of growth, profitability
  - Privately held
  - 50/50 Business Mix

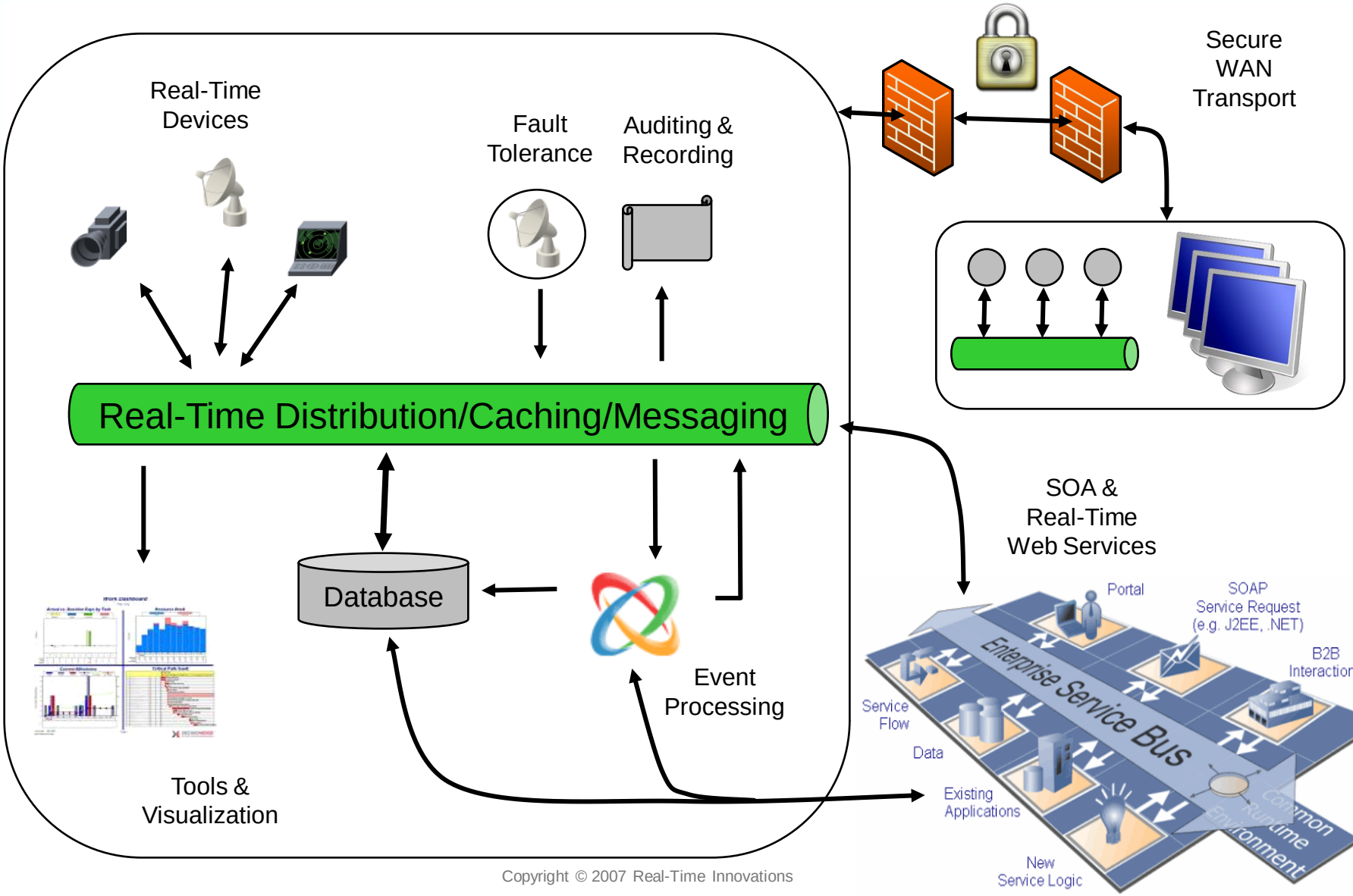
RTI Experience & Maturity

- 4th-generation product
  - Introduced in 1996
  - Current generation in use >2 years
- Proven in ~500 unique applications
- 98% customer satisfaction
- 100% customer recommendation

Use Case: Network Intrusion Detection



RTI's Integrated Infrastructure



Characteristics of event-driven systems

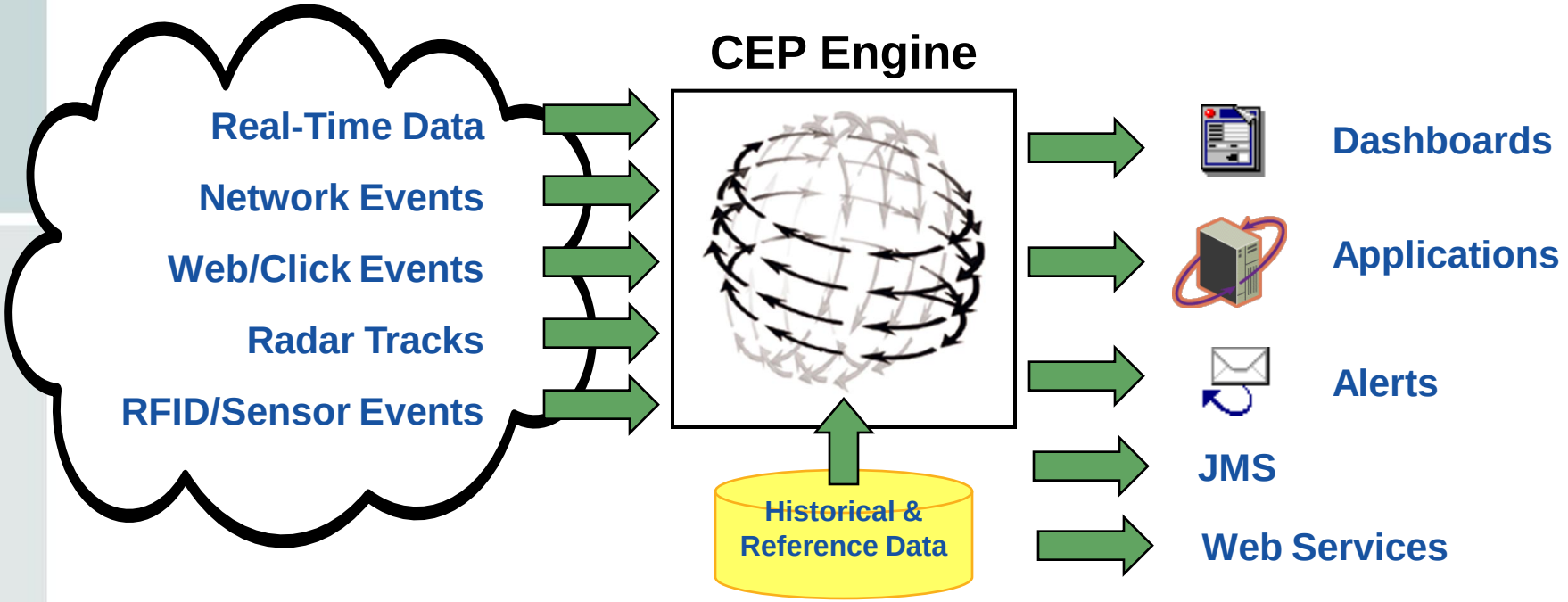
- Need to process data before enterprise-wide use
  - May need to eliminate duplicates, perform syntactic and semantic transformations on real-time data
  - Only some data from real-time systems may be relevant (example: alarm conditions)
- Data needs to be correlated
  - To generate meaningful event data, multiple *streams* and *topics* needs to be correlated
  - All data is not on DDS; need to correlate topics from other middleware and message-buses
- Lack of time window
  - Raw data may have short half-life; need semantics for time in the correlation queries

Characteristics of event-driven systems

- Need to continuously process data at high-throughput rates
  - Traditional correlation tasks require database persistence -- can lead to:
    - Persistence bloat (not all stored data will be useful)
    - Unacceptable application performance
- Code for event detection is heavy & static
  - Application code to infer events should adapt for short-development lifecycles
  - Developers for writing complex-pattern detection code need rich set of *filters* to infer events

Introducing Complex Event Processing (CEP)

- Useful for applications which:
  - Continuously process this data
  - Use high-speed data streams (100-1,000,000 msg/sec)
  - Support latency of one millisecond to seconds
  - Integrate multiple data streams and stored data



RTI Event Processing Interfaces

Packaged Adapters

- **RTI Data Distribution Service**
- Files
- Database
- MQ Series
- JMS
- TIBCO
- E-mail
- RSS/ATOM

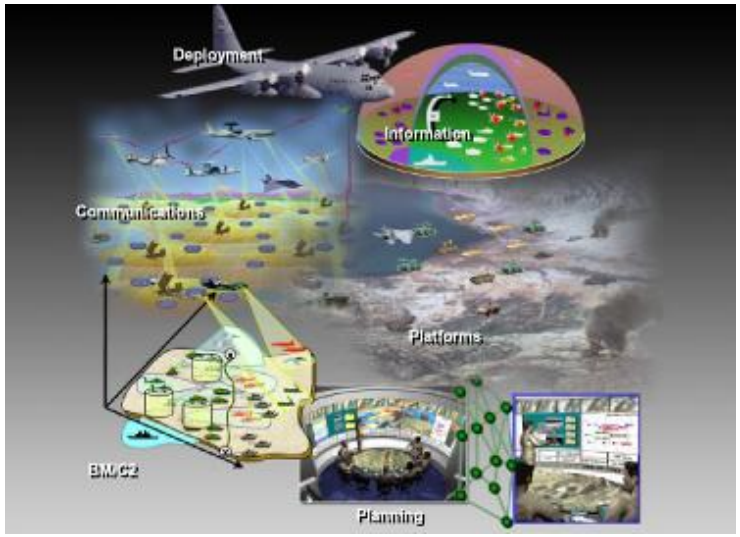
Protocol Adapters

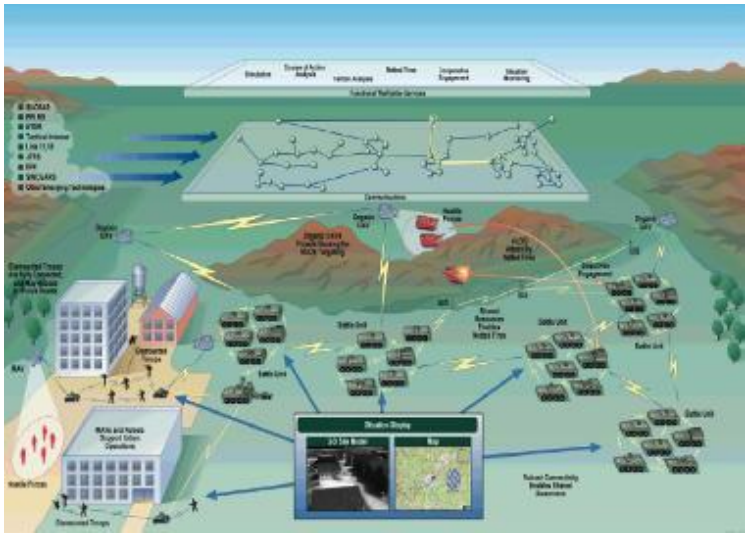
- Sockets
- SOAP
- SNMP

Languages

- C/C++
- Java
- .Net
- Perl
- Python

Introduction





- Many sources of events
- Inferences have to be made in real time
- Monitored events are not tailored to the problem we are trying to solve
- Events need to be correlated to build the information
- Casual and temporal tracking is not easy

Use Case: Event Processing in GIG

- The Global Information Grid (GIG) enables each operator to access quickly all relevant information.

- Challenges:
  - Scalability: Collect data generated by large-scale distributed sensor rich environments without interfering with communications.
  - Performance: Exploit the collected data without losing critical content, and overwhelming users
  - Data Relevance: Distributed entities have different concerns and perspectives
  - Changing Interests: New situations will require new monitoring queries and algorithms

Use Case: Event Processing in GIG

Approach:

- Use conditional monitoring
  - Continuous queries monitor Condition of Interests
  - Queries optimize flow of “significant news” to other nodes
  - Decoupling event detection and event-handling code enables monitoring for “changing interests”
- Do near-real time data analysis on local node
  - Limited number of joins
  - Leave data on generated nodes. Send processed (product) data over the network
    - Size of data “product” is several orders of magnitude less than the raw data size
  - Queries are mostly for counting and aggregations

RTI

Who uses CEP?

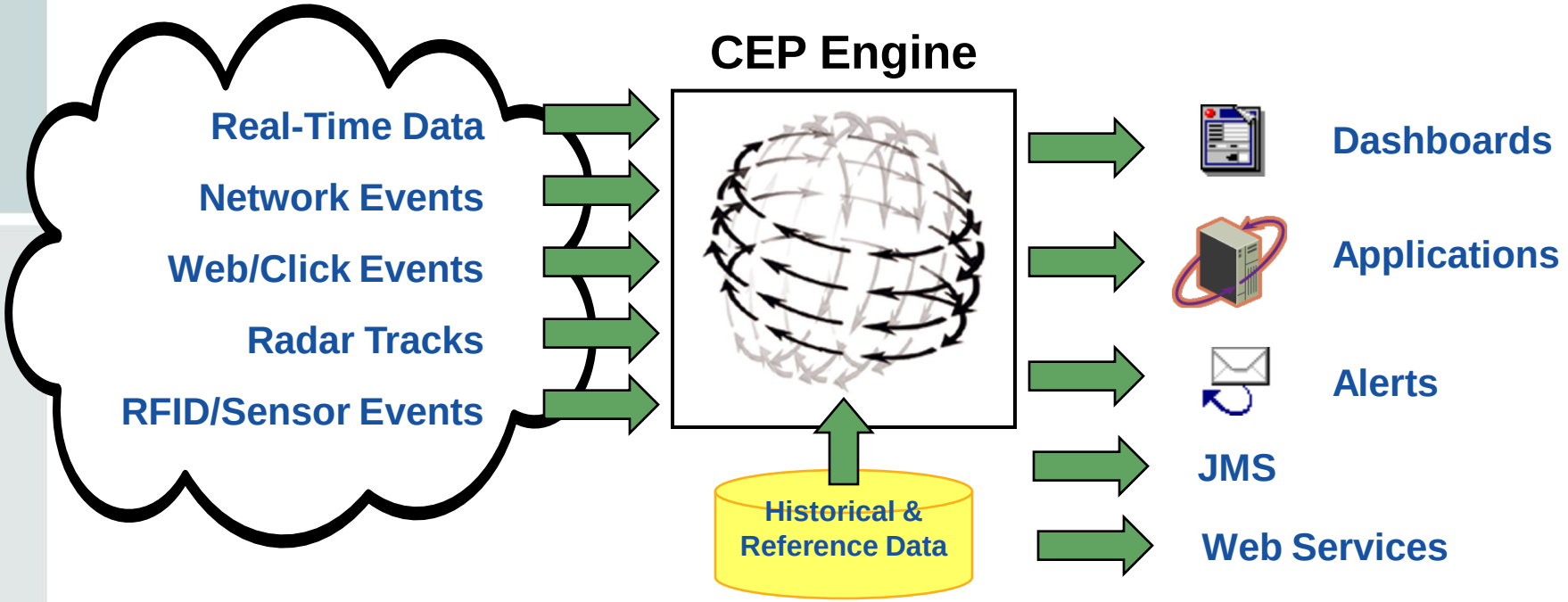
Used in diverse fields…and growing…

- Security:
  - Build filters for removing data to downgrade classification
  - Build patterns for detecting network intrusion (DARPA)
  - Perform predictive failure analysis
  - Monitor application and system performance
- Integration:
  - Bridge disparate middleware (DDS -> JMS)
  - Validate, cleanse, and **enrich** data during integration
  - Lower cost of integration to SOA-based systems
- Radar Systems:
  - Look for “erratic behavior” in radar tracks
  - Compare planned versus actual tracks

RTI

Introducing Complex Event Processing (CEP)

- Useful for applications which:
  - Continuously process this data
  - Use high-speed data streams (100-1,000,000 msg/sec)
  - Support latency of one millisecond to seconds
  - Integrate multiple data streams and stored data



Relational model semantics for CEP

Column	Datatype	Description
tradeid	Integer	a unique sequence number that identifies each trade
symbol	String	the stock symbol associated with this trade (such as MSFT, DELL or ORCL)
volume	Integer	the number of shares exchanged for this trade
price	Float	the price of each share exchanged for this trade

`StockTrade` Stream

timestamp	tradeid	symbol	volume	price
07:15:01	5001	MSFT	500	$29.00
07:15:02	5002	ORCL	1500	$18.00
07:15:03	5003	MSFT	1000	$28.00
07:15:05	5004	MSFT	1000	$31.00
07:15:06	5005	AAPL	500	$81.00
07:15:09	5006	AAPL	800	$82.00
07:15:13	5007	ORCL	500	$20.00
07:15:15	5008	AAPL	2000	$83.00
07:15:16	5009	MSFT	1500	$28.00
07:15:18	5010	AAPL	1000	$83.00
07:15:21	5011	MSFT	500	$29.00

CCL extends the SQL syntax to support continuous streams and the notion of time

Use Case	Continuous Computation Language (CCL) Queries

Windows & Aggregation	`INSERT INTO Crossed1GoingRight`
	`SELECT SquaresIn.color FROM SquaresIn`
	`KEEP 2 ROWS PER SquaresIn.color`
	`WHERE SquaresIn.x >= 50 AND PREV(SquaresIn.x) < 50 GROUP BY SquaresIn.color;`

Correlations	`INSERT INTO StuckInBoxEvent`
	`SELECT Crossed1.color`
	`FROM Crossed1, Crossed2,Crossed3`
	`WHERE Crossed1.color = Crossed2.color = Crossed3.color;`

Event Pattern Matching	`INSERT INTO StuckInBoxEvent`
	`SELECT Crossed1.color`
	`FROM Crossed1, Crossed2, Crossed3`
	`MATCHING [5 seconds : Crossed1, !Crossed2 && !Crossed3]`
	`ON Crossed1.color = Crossed2.color = Crossed3.color;`

DDS complements CEP in real-time systems

- Data Distribution Service (DDS) is about sending and receiving data in distributed systems.
  - Usually, data has real time characteristics with low latency & high throughput needs
- CEP is about detecting events from streaming and/or real-time data

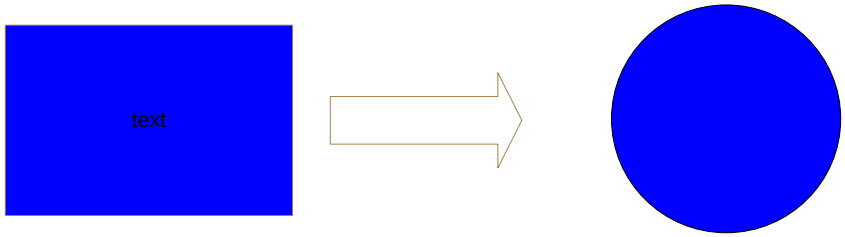
---

- DDS ideal for CEP systems with real-time needs:
  - Rich set of QoS to maximize system performance
  - *Helps scale CEP processing* -- Typically performs orders of magnitude better than broker-based implementations of JMS
  - Typically supports more RTOS vendors than other implementations.

- CEP ideal for real-time systems for detecting events:
  - Data is not persisted before query. Can manage typical low-latency and high-throughput constraints of DDS networks

RTI

CEP Transformation Demo

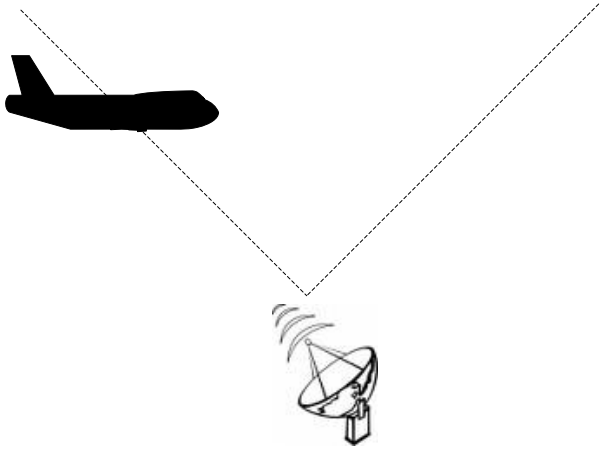


This demo illustrates:

-Ease of use in integrating different middleware

-Ability to do real-time processing during integration

CEP Radar Tracking Demo



This demo illustrates:

- Ability to do time-based pattern matching
- Using CEP to infer “relevant events” from raw-data

CEP helps SOA in event-driven systems

- SOA is about interoperability. CEP is about analyzing large volume of real-time events.

- CEP helps SOA systems by:
  - Reducing the cost of managing event-*detection* code
  - Addressing data-rate impedance mismatch between real-time systems and the enterprise

Use CEP to orchestrate invocation of different web-services

RTI

Do I need CEP?

Yes, if there is a need to continuously process real-time data

- Do I have real-time data?
- Is there a process for the information that I need to know *now*?
- Do I need to find complex & enriching patterns, computations & correlations in real time?
- Do I need to bridge different streaming technologies (including RTI Data Distribution Service) in real time?

Thank You!

For more information:
Visit http://www.rti.com/

Start your CEP evaluation today:
info@rti.com