

# XMessages: Building Reliable, Persistent Messaging Middleware for the Grid

---

Yogesh L. Simmhan  
Aleksander Slominski, Dennis  
Gannon, Mathew Farrellee,  
Albert Rossi, Octav Chipara  
*Extreme! Computing Lab  
Indiana University*

## Grid

---

- Groups of collaborating users
- Share collective access to resources

## Grid Services

---

- Security
- Remote execution
- Information services
- Data services
- Monitoring

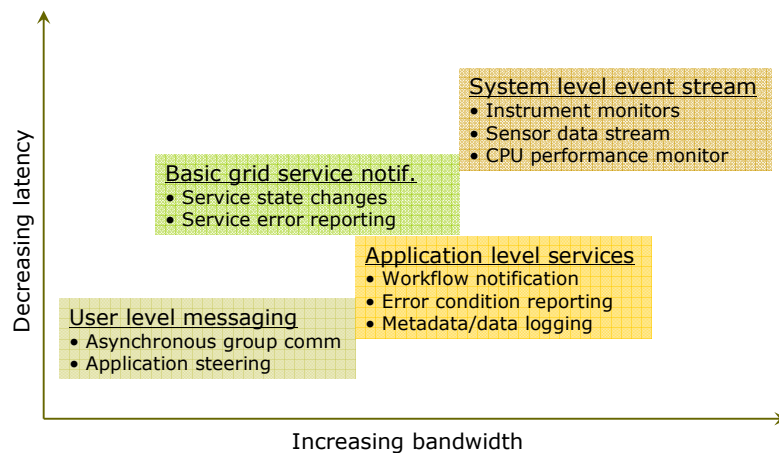
## Grid Messaging & Notification

---

- Basic service level
  - Changes in state of service
- System level
  - Hardware performance monitor
- Application level
  - Workflow messaging
- User level
  - Application steering

## Messaging Requirements: Bandwidth vs. Latency

---



## Messaging Requirements (contd.)

---

- Reliability
- Security
- Location independence
- Persistence
- Scalability

## Existing Messaging Systems

---

- CORBA notification
- ECho event delivery system
- Java Messaging System (JMS)
- Narada Brokering

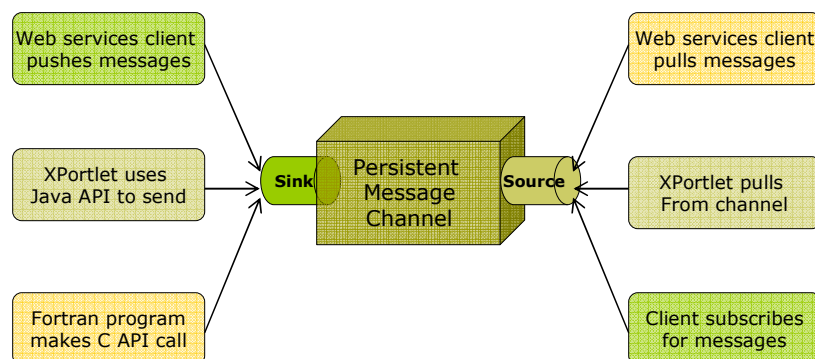
## XMessages: Key features

---

- Arbitrary XML messages
- Reliability
  - Publisher and listener agents
- Persistent storage
  - RDBMS, Native XML DB
  - Historical message retrieval
- "Push" and "Pull" message notification
  - SQL, XPath queries

## XMessages Architecture

---



## XMessages API

---

### □ *XMessageSink*

- void handleMessages  
( XMessageBatch messages )

### □ *XMessageSource*

- XMessageBatch requestMessages  
( XMessageRequest request )

## XMessages API (contd.)

---

### □ *XMessageBatch*

- Array of XMessages
- Tokens – soft state

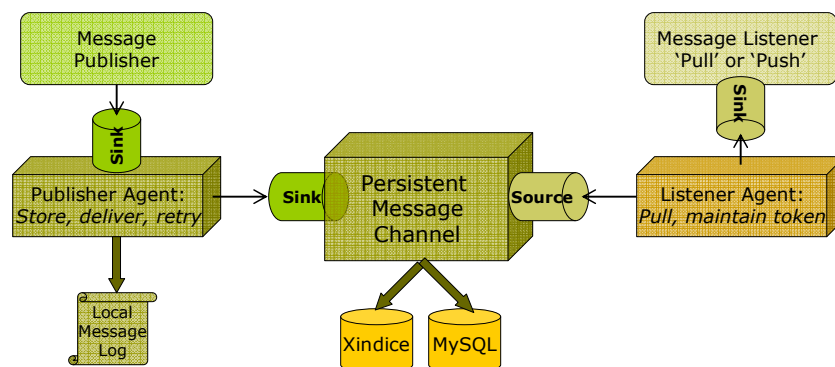
### □ *XMessageRequest*

- Query/token
- Max. messages, time-range

## XMessages Implementation

- ❑ Client side 'Agent' library
- ❑ Wrapper around channel
- ❑ Publisher agent provides reliability
- ❑ Listener agent allows "pull" and "push"
- ❑ Implementation
  - Java and C/C++ publisher agent
  - Java listener agent

## XMessages Implementation (contd.)



## Publishing Messages

---

### □ Client

- Get reference to channel
- Create publisher agent for channel
- Call `handleMessages` on agent with the message

### □ Publisher agent

- Buffer and log message to store (File, DBMS)
- Try to send
- If success, mark in log. Else retry sending.

## Retrieving Messages

---

### □ Client

- Get reference to channel
- Create listener agent for channel
- Get "pull" or "push" subscription from agent based on *query*
- If "pull", call `pullMessages` on agent. Else implement `handleMessages`
- Renew lease

## Retrieving Messages (contd.)

---

### □ Listener agent

- If “push” subscription, start thread to pull from channel using *query*
- Call `handleMessages` on client when messages have been retrieved from channel
  
- If “pull” subscription, wait for user to invoke `pullMessages` on agent
- Call `requestMessages` on channel with query or token. Return result to client.

## Message handling in the Channel

---

### □ When `handleMessages` is called, store message in database

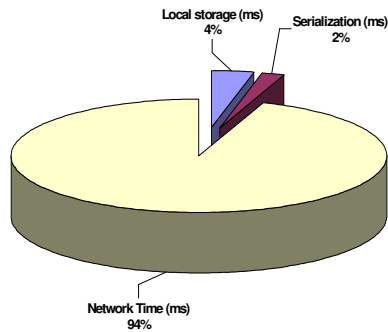
- Data, meta-data stored separately
- Xindice, MySQL, HypersonicSQL
- RDBMS vs. Native XML

### □ When `requestMessages` is called

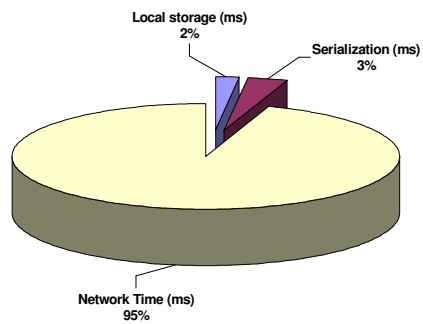
- Extract query from token
- Execute query on database
- Return messages and next token

## Performance

### □ Performance of publisher agent



Time distribution when 1,000 messages of size 1K were sent in batches of 10



Time distribution when 10,000 messages of size 1K were sent in batches of 1,000

## Future work

- Add security layer
- Bridge to other messaging systems
  - JMS
  - Narada
  - OGSA
- Plug-in architecture

# Questions

