Grid Messaging System

Wojciech Czech

AGH University of Science and Technology, Kraków

Openlab Summer Student Program

30th August 2007

Outline

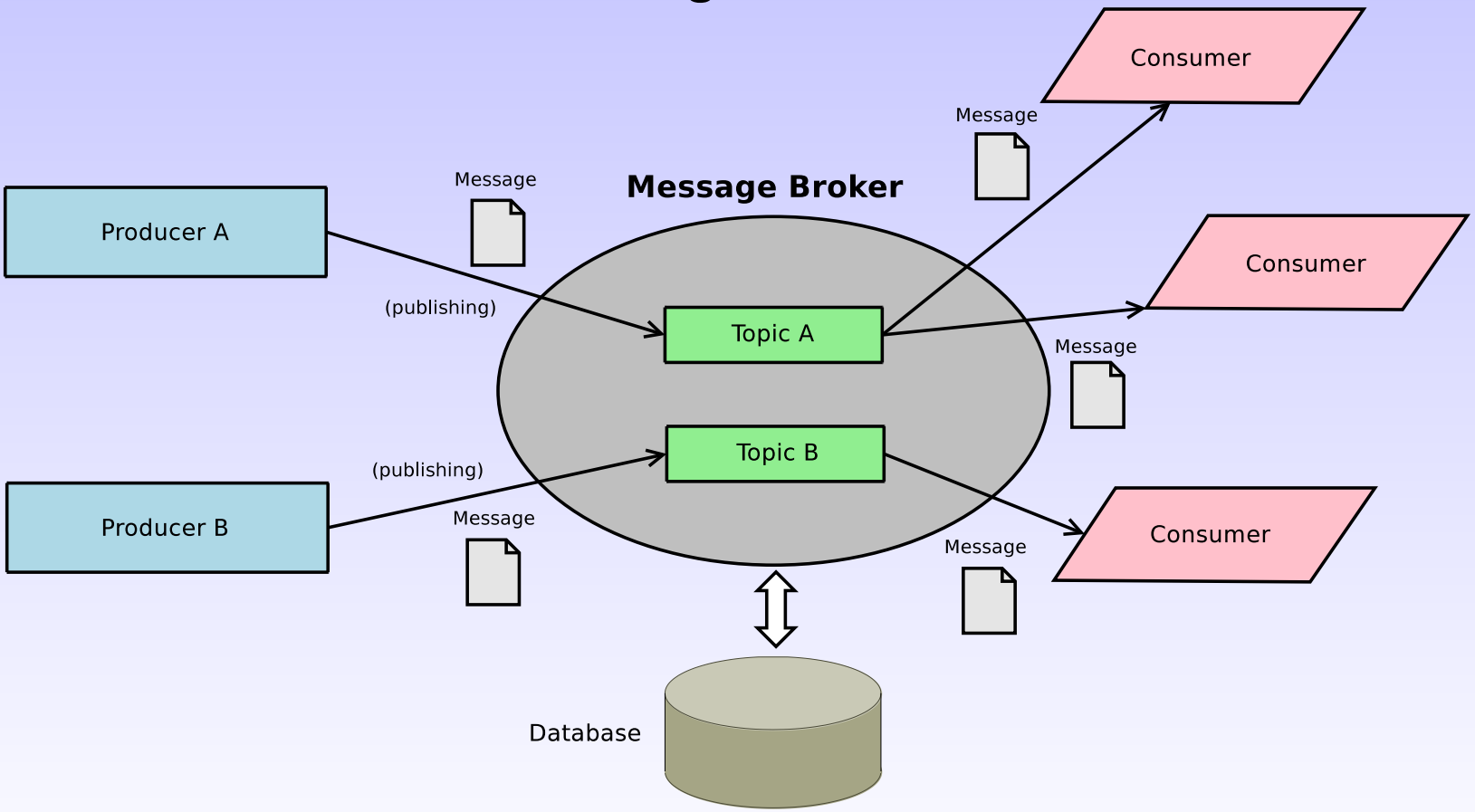
- Introduction
- Message brokers - Active MQ
- Performance tests
- Integration with SAM
- On going work - security
- Future work

Goals

- Create prototype of transport layer for Grid monitoring systems: SAM, GridView, Fabric Monitoring
- Follow directions indicated by Grid Monitoring Working Group
- Implement Grid Monitoring Probe Specification
- Provide interoperability, portability, scalability and reliability of message passing
- Taking advantage of existing tool - avoiding new development work

Idea

Using publish-subscribe schema supported by Active MQ message broker



MOM, JMS & Message Brokers

- Message-Oriented Middleware (MOM) - infrastructure that allows for passing asynchronous, event-driven messages
- Message Broker - intermediate software that forwards messages from sender to receiver and performs some additional processing such as translation between various message formats, storing in database or applying filters.
- Java Messaging Service (JMS) - specification of reliable, asynchronous, loosely coupled communication; common API - various implementations

Active MQ

Active MQ is existing solution, widely used in industry and research. It implements JMS, provides translations between various protocols and supports:

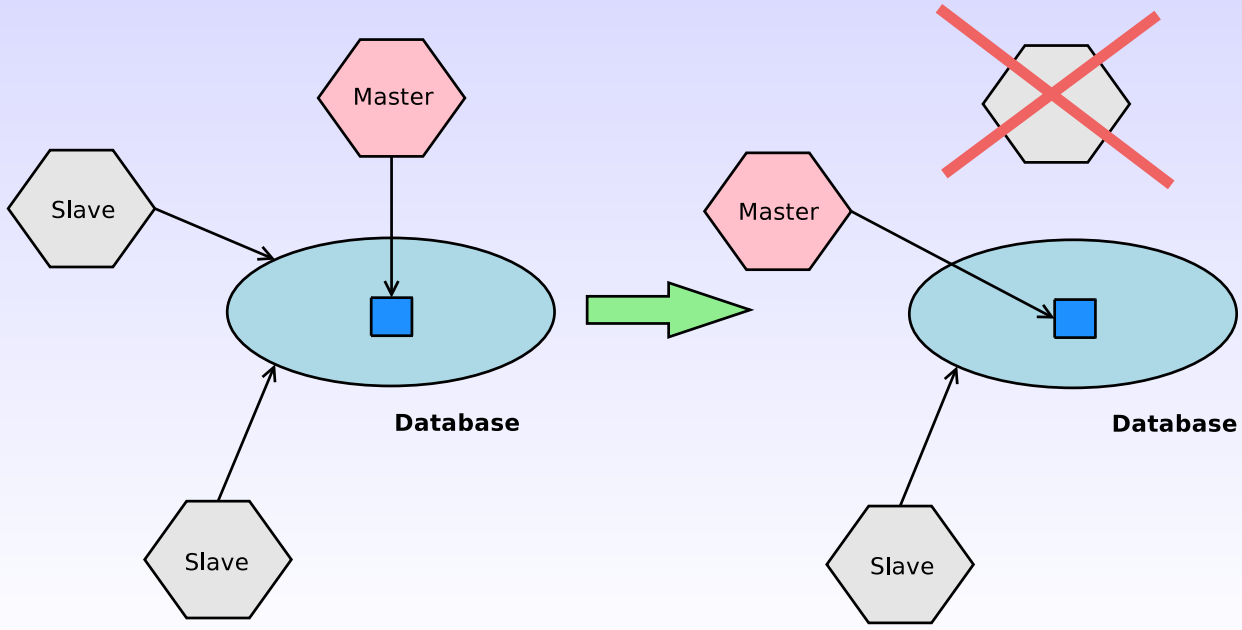
- OpenWire protocol (Java, C, C++, C#)
- STOMP protocol (Java, C++, C, Ruby, Perl, Python, PHP, ...)
- Master-Slave mechanism (fault tolerance)
- Networks of Brokers (scalability)
- Message Groups (flexibility)
- Transport protocols: in-VM, TCP, SSL, NIO, UDP, multicast, JGroups and JXTA transports
- Persistent messaging using JDBC and High Performance Journal

Persistent messages

- Durable subscriptions allows for receiving messages published when subscriber was not connected to Message Broker. Such messages are stored in database and then delivered to subscriber if it reconnects.
- Via JDBC various databases can be used (e.g. MySQL, Oracle or Postgres).
- AMQ can use journaling to improve performance of data storing.
- Durable subscriptions are used to provide subscriber fault tolerance (messages cannot be lost due to subscriber failure).

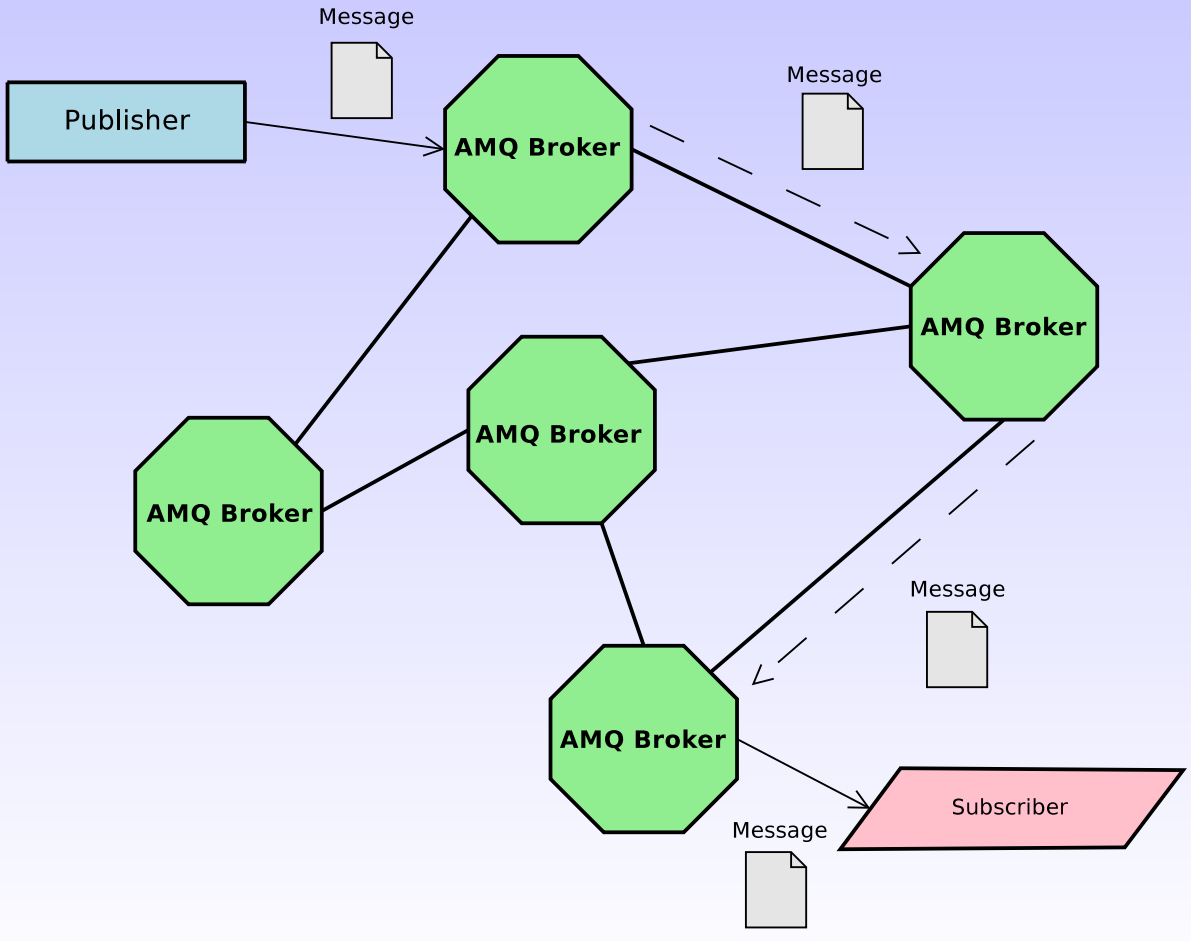
Reliability: Mater-Slave, dynamic failover

- Pure Master-Slave (Slave with full replication of Master state)
- Shared File System Master-Slave (Exclusive locks in shared broker data directory to provide mutual exclusion)
- JDBC Master-Slave (Exclusive locks in database)
- Dynamic failover - allows for client-side, dynamic change of broker after connection failure



Scalability: Networks of Brokers

Networks of Brokers mechanism provides scalability by enabling distributed queues and topics



Scalability: Networks of Brokers, brokers' discovery

- The network of AMQ brokers can be treated as IP network, where brokers are similar to routers.
- Network of Brokers connections can be configured similarly as routing tables (setting TTL's, per-topic routing rules, etc.)
- The publisher and subscriber can be connected to different brokers and the message will be forwarded and delivered accordingly.
- Networks of Brokers do not provide full fault tolerance but owing to brokers redundancy we obtain high availability.
- Client can connect to network by static list of brokers' URL's, via multicast discovery agent or with the use of Zeroconf discovery.

Interoperability: STOMP protocol

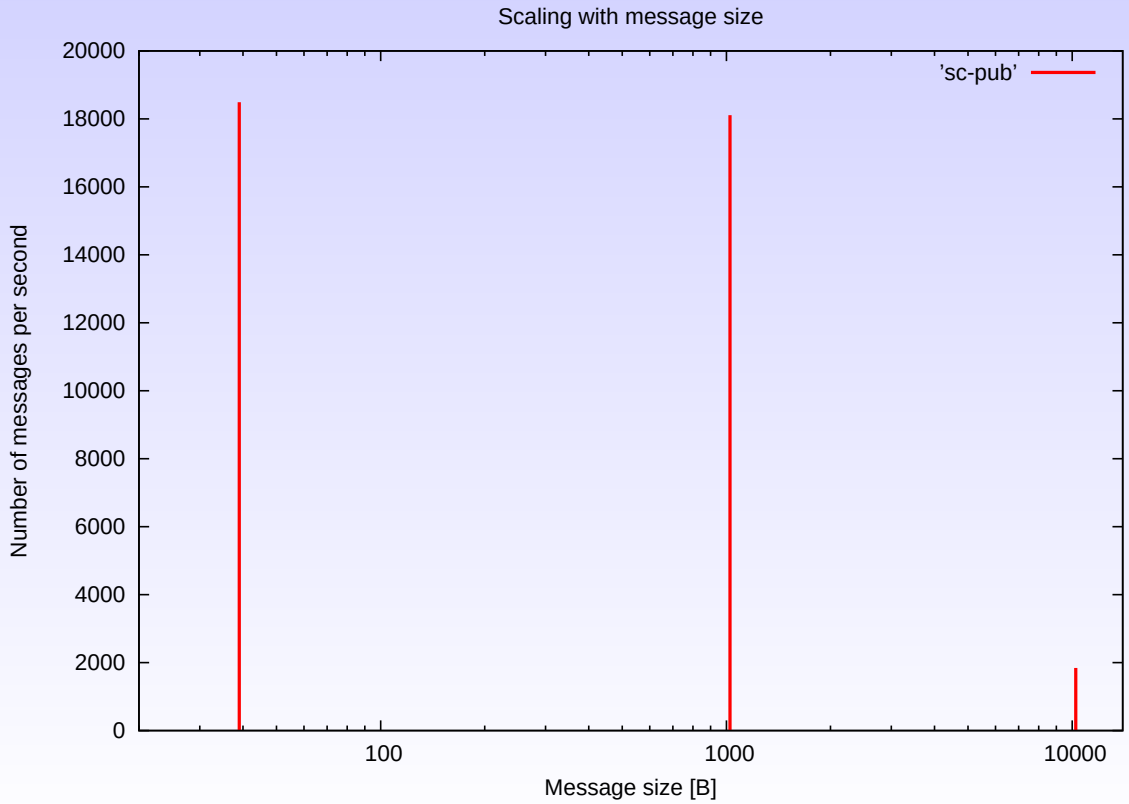
- Streaming Text Orientated Messaging Protocol (STOMP) introduces easy text-based message format and allows for communication any STOMP client with any STOMP broker.
- Active MQ implements STOMP broker therefore allows for queue/topic communication between many various clients e.g., Java JMS, Java STOMP, Java OpenWire, Python STOMP, Perl STOMP, C STOMP, Ruby STOMP etc.
- STOMP message consists of two parts: headers and body.
- Headers section can contain many extensions which enables JMS features such as durable subscriptions, message selectors SQL 92, synchronous communication, acknowledgments etc.

Active MQ - performance tests

- In order to evaluate capabilities of broker in different communication scenarios we performed some tests, trying to investigate number of messages broker can forward per second.
- The experiments consisted of following parts
  - Single publisher/single consumer - evaluation of maximal performance
  - Scaling of performance with message size
  - Scaling with number of producers
  - Scaling with number of topics
- For testing purposes we used Active MQ 5.0 running inside Tomcat 5.5 and clients written in Python, Java or using `netcat` to write messages directly to socket.

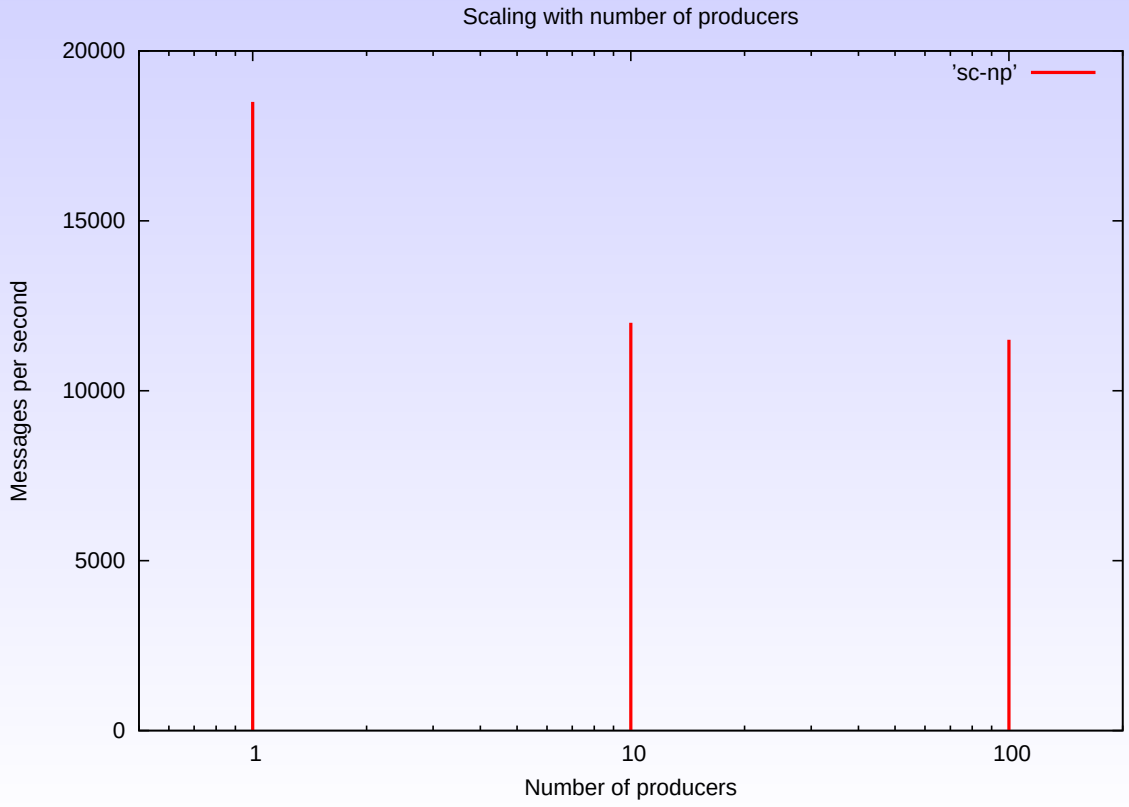
Performance tests results (1)

- Maximal performance: 18500 messages per second
- Scaling with message size (logarithmic x-axis scale)
- Lower performance for 10KB message size - maximal throughput of Ethernet link reached



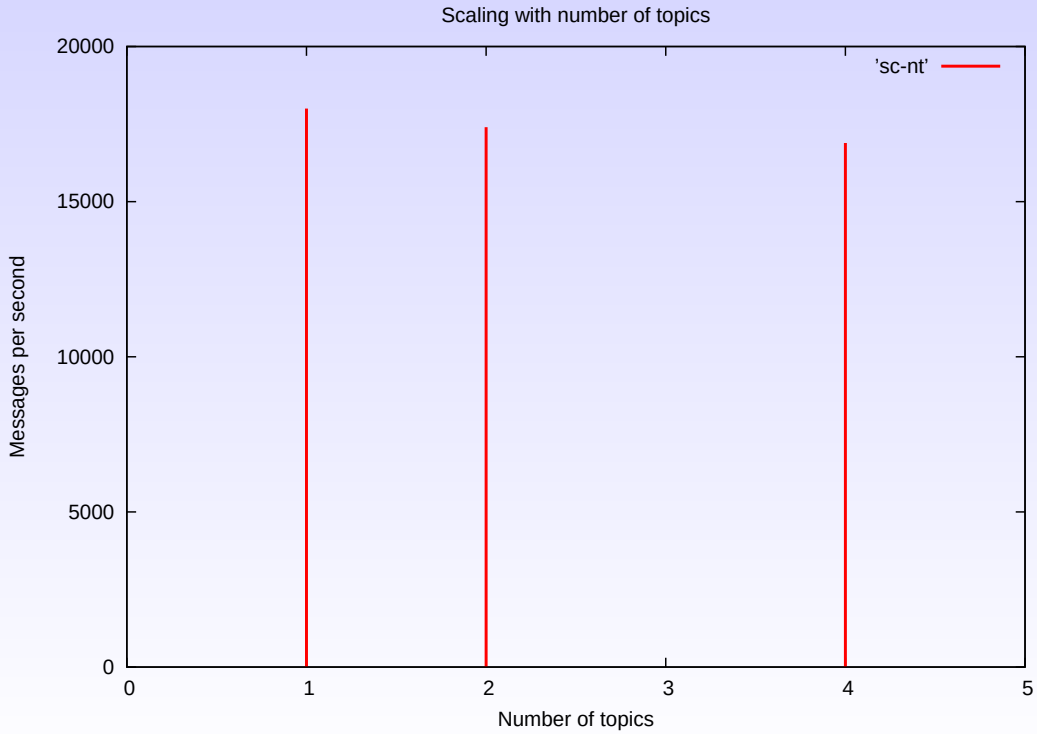
Performance tests results (2)

- Scaling with number of producers (logarithmic x-axis scale)
- Configuration: one consumer, n producers
- With increasing number of producers performance does not deteriorate significantly

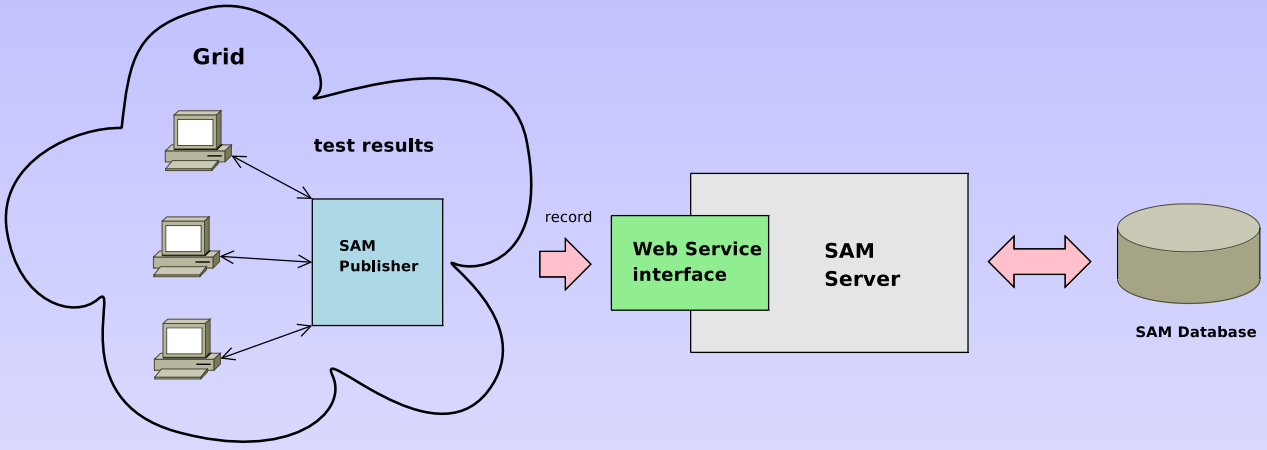


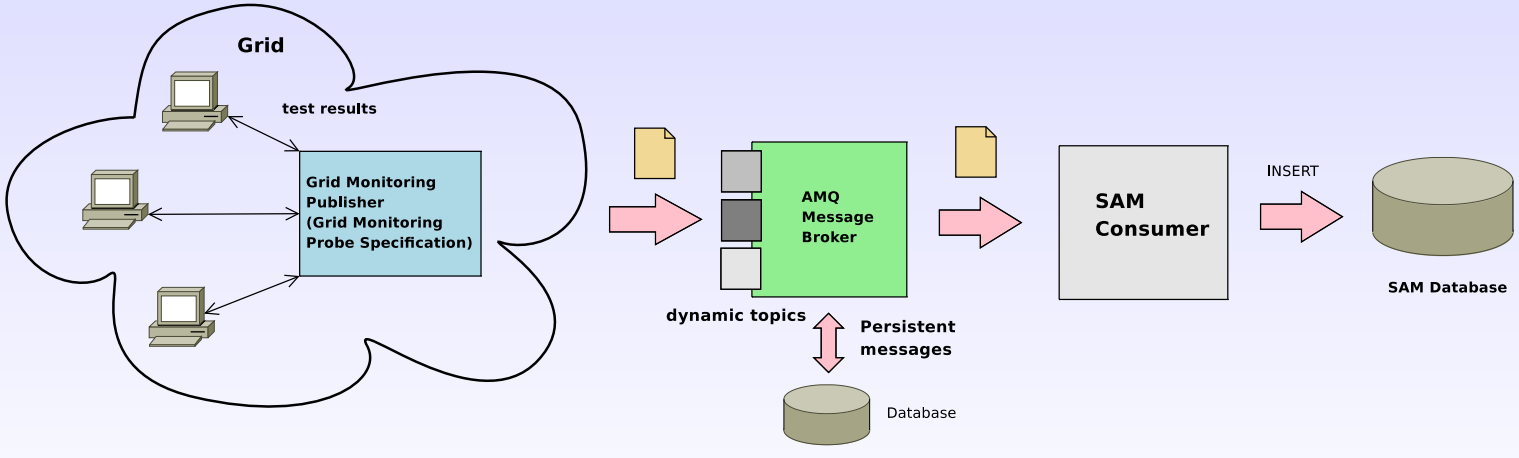
Performance tests results (3)

- Scaling with number of topics
- Configuration: n publisher-consumer pairs, n different topics
- y-axis: sum of messages received per second by each consumer



Integration with SAM





Benefits of new approach

- No single point of failure (Web Service).
- Protection against broker failure (Master-Slave mechanism) and consumer failure (durable subscriptions).
- Flexibility - new consumers can be added easily.
- Data replication by adding new consumer and database.

Grid Monitoring Publisher

- Written in Python, uses simple STOMP library
- In conformity with Grid Monitoring Probes Specification (James Casey) and Grid Publisher Specification (draft by Piotr Nyczyk)
- Information contained in GMPS messages have form of attribute-value sets. Part of such messages can be additionally attached to STOMP message headers, what enables subscription selectivity.
- Grid Monitoring Publisher performs validation of messages before sending them to broker and is configurable in terms of message destination, message aggregation, additional headers, etc.

SAM Consumer

- Java application running inside Tomcat
- Durable JMS subscriber
- Adapts GMPS messages sent by Grid Monitoring Publisher to form suitable for SAM database schema
- Performs some processing to get attributes required by SAM but not included in GMPS messages
- Uses GridView table handlers
- Can be adapted to support Grid FTP logs and job monitoring (Grid View)

Testing prototype with SAM

- SAM Submission Framework was adapted to use new publisher (as alternative).
- SAM Consumer received messages published by Grid Monitoring Publisher and translated them to SAM format.
- Full scale tests for sensors: LFC, SRM and SE (more than 200 sites, all tests from SAM UI).
- Sensors CE and gCE - tests to be done (CERN firewall modification needed - waiting for approval).
- To do - test of persistent messaging.

On going work - security

Channel security vs. message security

1. Signing
   - User proxy certificates - long path of verification.
   - Signature attached to message body.
   - Public part of proxy certificate attached to message body of first message in sequence to avoid redundant increase of message size. The certificate is cached by consumer. The successive messages contain only hash of certificate.
   - Additional message headers indicating signature and/or encryption.
2. Encryption - using public key of consumer application

Future work

- Finish tests with SAM (CE, gCE and durable subscriptions)
- Migration of existing grid monitoring systems (SAM, GridView) to use new transport
- Design and deployment of network of brokers (e.g. a broker per ROC etc.)
- Finish implementation of message authentication (signatures, encryption)
- Consider authorization methods

Thank you for your attention!