Comparing Solace's Appliance-Based Guaranteed Messaging with Software Brokers

Guaranteed messaging, also known as persistent messaging in the JMS world, is a quality of service whereby publisher applications send a message and are guaranteed it will be delivered to all consuming applications even if they are unable to receive the message at that time.

Guaranteed messaging solutions temporarily persist published messages until receipt by all destinations is acknowledged. Software-based message brokers must persist messages in external storage systems. The limitations of storage technology available now and for the foreseeable future cause undesirable behaviours and impose serious performance and real-time limitations

This paper, intended for IT professionals already familiar with the basics of guaranteed messaging, JMS messaging and Solace technology, compares software based message brokers with Solace message routers. It describes how the implementations of these products affect their ability to provide the features required by today's enterprise.

Solace Systems®

http://www.solacesystems.com

Table of Contents

1 Introduction 3

2 The Need for a New Breed of Guaranteed Messaging Technology 4
2.1 Market Trends 4
2.2 Industry-Specific Use Cases 4

3 Description of Guaranteed Messaging Systems 4
3.1 Software Broker Implementations 5
3.1.1 Persistence 5
3.1.2 Acknowledgement 6
3.1.3 Delivery 6
3.1.4 Dequeueing 6
3.2 Description of Solace Messaging System 7
3.2.1 Persistence 7
3.2.2 Acknowledgement 8
3.2.3 Delivery 8
3.2.4 Dequeueing 9

4 Enterprise Guaranteed Messaging Requirements 9
4.1 Performance 9
4.2 Client Isolation 9
4.3 Redundancy and High Availability 10
4.4 Disaster Recovery 10
4.5 WAN Distribution 10
4.6 Scalability 10
4.7 Management 10
4.8 Total Cost of Ownership 11

5 Case Study – Online Retailer 11

6 Summary 12

1 Introduction

Software-based message brokers rely on third-party storage technology to persist messages and therefore suffer a number of undesirable behaviours resulting from performance limitations of the underlying disks. Disk based storage technology has not enjoyed the performance increases that Moore's Law has brought to microprocessors, and the bottleneck in guaranteed messaging is storage technology, not processing power.

This causes something called the "I/O gap" that continues to widen, making it difficult for software products that rely on storage to realize performance gains. A related example is the performance of in-memory data grids versus traditional database systems. Solid state storage technology addresses some of the performance issues related to rotating media, but those improvements come at a significant cost when redundancy is a requirement.

Solace message routers are built with patented hardware technology that optimizes the execution of guaranteed messaging. This paper shows how Solace's message routers system offers capacity, performance and robustness that software-based solutions just can't match, focusing on how Solace addresses the following problems:

Problem	Solace's Solution
Disk performance increases are not keeping up with processor improvements.	Replace disk based storage with patented hardware (bridging the IO gap).
Disk accesses have long and unpredictable latency limiting real-time system performance.	Remove all disk access from the real-time processing path.
HA failovers are slow because the backup system must read message data and delivery state from disk before it can resume providing service to applications.	Synchronously mirror stored message data and delivery state to a backup system in hardware so the backup system is always kept up to date, enabling fast HA failovers regardless of how much data is stored.
Applications are un-necessarily complicated because software message brokers are slow to return acknowledgements limiting throughput.	Persist messages to non-volatile RAM (not disk) which results in low latency responses to publish events.
Message broker performance is unpredictable and typically degrades when slow subscribers are present.	Perform disk accesses (and associated unpredictable latency) in a background thread so well-behaved message flows are unaffected.

It's important to acknowledge the magnitude to which these limitations affect the performance of software based message brokers. For example, in lab environments software-based message brokers with persistence can demonstrate performance of 6-20,000 messages per second, while Solace message routers show performance of over 450,000 messages per secondwith fan-out rates up to 1.6 million messages per second. In demanding production environments, many developers actually limit each software broker to 2-3,000 guaranteed messages per second because the brokers can't reliably handle more than that in real world traffic situations. And when applications require messages to be synchronously replicated to a remote disaster recovery site using SAN replication, the performance of software-based brokers degrades even more.

Solace message routers offer an enterprise class, multi-tenant, guaranteed messaging solution with performance, scale, stability and manageability that frees application developers to focus on the business requirements of their applications rather than the idiosyncrasies of their chosen messaging system.

By offering 1-2 orders of magnitude more throughput in a compact device, Solace lets companies dramatically reduce OS licensing costs, power consumption, rack space, management overhead, application complexity, software licenses, and system fragility.

2 The Need for a New Breed of Guaranteed Messaging Technology

2.1 Market Trends

Today's enterprise applications are placing more demands on middleware infrastructure. Trends seen across a wide range of industries include:

- **Data Volumes:** The amount of digital information in the world is doubling every two years (according to IDC)
- **Real-Time Analytics:** Real-time analytics are being demanded by an increasing number of applications
- **Regulations:** Regulatory requirements are dictating the archival of increasing amounts of data
- **Big Data:** The desire to analyze "big data" drives the need to warehouse massive volumes of information
- **Global Computing:** The global nature of today's enterprise requires data to be moved across the WAN

2.2 Industry-Specific Use Cases

These challenges are faced by CIOs across many verticals. For example these trends have been observed in the following industries:

- **Capital Markets:** Middle office systems are taking ever increasing volumes of order/trade data from the front office for many asset classes, and distributing that data to more and more systems for analytics, settlement, data warehousing, order status monitoring, risk management and regulatory reporting.
- **Telecommunications:** Service providers are attempting to process more and more usage information per service per client in near-real-time for faster billing, customer analytics, customer satisfaction and big data storage
- **Internet Commerce:** More websites are monitoring and archiving application logs and clickstream events in near-real-time so they can analyze everything from system performance to user behavior and experience.
- **Online Gaming & Gambling:** Real-time responses to an ever-growing user population with more and more applications needing this information for in-game analysis in addition to the e-commerce requirements above.

The software-based solutions currently available were not designed to meet the requirements of current and next generation applications. A new disruptive technology is required to get maximum benefit from advances in data analytics, in real-time and in post processing across a global enterprise.

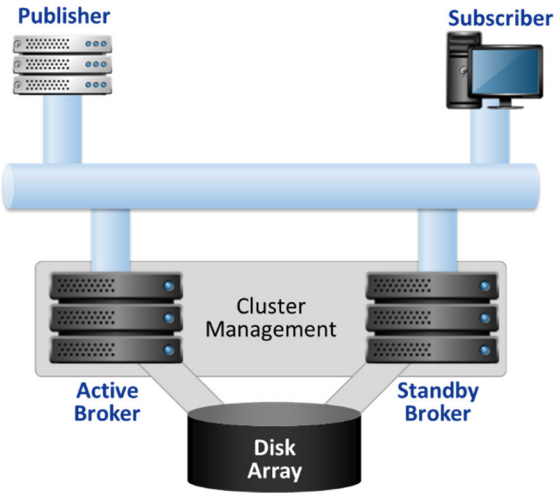
3 Description of Guaranteed Messaging Systems

All guaranteed messaging systems must perform the following tasks: persist and enqueue messages, acknowledge the publisher, deliver to consumers and dequeue upon reception of a consumer acknowledgement. The differences between implementations are in how they perform these four tasks.

- **Persistence:** Persisting a message involves receiving it from a publisher, looking up its destinations, and storing the message and its destinations in non-volatile storage.
- **Acknowledgement:** Once the message is safely in non-volatile storage, the message broker can return an acknowledgement to the publisher, stating that the message is stored and will be delivered to all subscriber destinations.
- **Delivery:** The message is delivered by the message broker to subscriber destinations.
- **Dequeueing:** The subscriber endpoints return an acknowledgement to the message broker, acknowledging that they have successfully received the message and that it is okay for the message broker to delete the message. Once the message broker has received an acknowledgement from all subscriber destinations for a given message, it can delete the message from non-volatile storage.

3.1 Software Broker Implementations

This diagram represents a typical guaranteed messaging system consisting of two software based message brokers running on commodity servers.



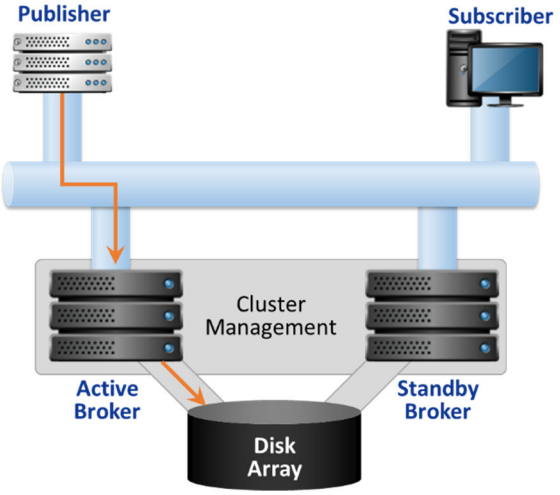
This example contains two message brokers for redundancy; one active and one standby. Other implementations may support clustering and N+1 sparing which controls costs in situations where many software brokers are deployed due to performance limitations of the individual brokers. The cost benefits of clustering and N+1 sparing are paid for with increased HA failover times because a newly active broker must reload all state before activity can resume. In this example both message brokers are connected to a shared disk array, either directly or via a SAN.

The software brokers are running some form of cluster management software to ensure consistent access to shared storage on the disk array and for health monitoring. There are two clients of the messaging system, one is a publisher and one subscriber.

3.1.1 Persistence

The publisher sends a message to the active message broker, which looks up the message's destination(s) and stores the messages to disk.

Message payload and delivery status for all destinations must be stored on disk, because in the event of a failure the standby will require this information.



Operating systems will attempt to cache files in RAM to improve performance. The file system cache held in system RAM is volatile and must be flushed when writing messages to disk. This must be completed before an acknowledgement is returned to the publisher because A) data in the cache is volatile and B) data in the cache is not available to the standby in case of failure. Flushing the cache to disk introduces latency (even if the disk is equipped with write back cache) and unpredictable performance.

Latency in the feedback loop from publisher send to the return of an acknowledgement is a crucial parameter because it determines the rate at which a publisher doing blocking sends, as is the case with JMS publishers, can publish. While it is true that some messaging systems are able to run in a mode that does not flush the file system cache to disk to improve performance; this is an unfortunate trade-off between reliability and performance that accentuates the fact that storage is the main bottleneck and often such potential of data loss or corruption is not an acceptable business trade-off for increased performance.

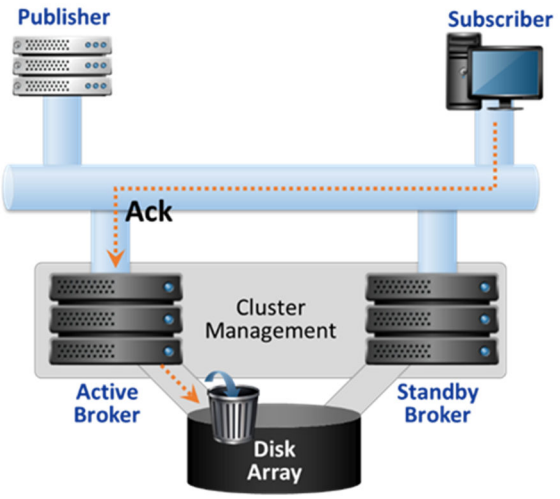
Storage must be external and shared for redundancy, possibly using special cluster file systems included with cluster management suites (additional software licenses required).

Such systems typically use direct attached storage (DAS) instead of network attached storage (NAS) because differences between storage vendors (such as how they handle synchronous writing of files and application fencing requirements) can introduce data corruption such as split brain situations where both brokers think they are active.

Replication over a WAN for disaster recovery (if required) increases write latency delaying the return of acknowledgements to the publisher and decreasing the broker's write performance which affects all applications.

3.1.2 Acknowledgement

Once the message is stored on the disk, the message broker returns and acknowledgement to the sender. For many applications, such as those using the JMS API, the round trip time from publisher send to receipt of this acknowledgement determines the rate at which the publisher can send messages.



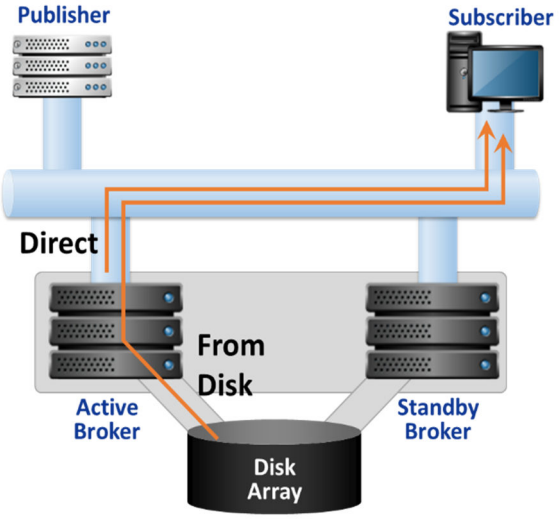
Operating system interactions on the message broker introduce variability in latency.

Disk throughput can be optimized by bundling writes but, this increases latency; the file system cache still needs to be flushed before the acknowledgement can be returned.

To combat these effects and to allow applications to make forward progress in the face of a message broker that is slow in returning acknowledgements; applications can be designed to make use of multiple connections and threads. This added complexity would be unnecessary if the broker was simply faster.

3.1.3 Delivery

After the message has been persisted to disk and the publisher has received acknowledgement, the message is forwarded to relevant subscribers. Software brokers typically cache a copy of the message in RAM for some period of time so that it does not need to be retrieved from disk if the subscriber is online.



Messages that are unable to be delivered immediately due to slow or offline subscribers, however, must eventually be retrieved from disk.

Disk write performance can be optimized; the width of the RAID stripe can be adjusted to increase bandwidth so as long as the write cache on the disk controller is large enough the controllers can efficiently optimize the head movement. Read latency, on the other hand, is hard to accelerate because of the seek time associated with disk head movement. Message data is typically short lived which frustrates read caching algorithms. Reading messages from disk also consumes resources on the disk controller that would otherwise be used to write messages to disk affecting the publish rate.

3.1.4 Dequeueing

The message broker waits for an acknowledgement from the subscriber and updates the delivery state for the message. If all destinations have acknowledged receipt of the message, it can be deleted from disk.

Delivery state of the message is updated on disk so that the standby system will have access to the delivery status in case of failure. The typical block size of a modern file system is 4kB; that is the minimum unit that the file system can operate on. Due to the relatively large block size, small updates (such as processing a subscriber acknowledgement) are very inefficient. These types of workloads are better suited to storage technologies that are accessible on a more granular level (such as DRAM) however the requirements for these updates to be both non-volatile and accessible by the standby broker makes the external disk array the only option available to the software message broker.

File systems do not strictly need to be flushed after updating the delivery state of a destination but the likelihood of a duplicate delivery after a failure of a broker increases if it is not.

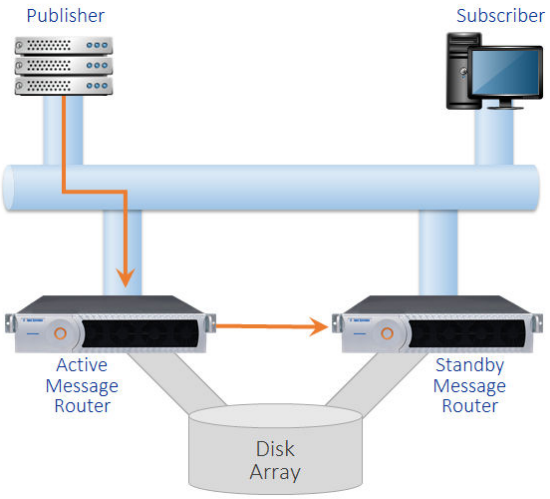
Performance related issues affecting software message brokers are largely caused by their reliance on disk based storage. New technology in the form of solid state storage addresses some of the issues but does not provide a complete solution. For example SSDs in the form of PCI Express cards provide much better performance and eliminate the problem of having to move disk heads in order to read a file but do not allow for a redundant solution. These products leave data trapped inside of the server hosting the message broker software in the case of a hardware failure. All flash array products solve the read problem and provide a redundant solution but are expensive.

3.2 Description of Solace Messaging System

This diagram shows a typical deployment of a pair of Solace message routers for high availability. As with the previously described software broker there are two clients one publisher and one subscriber. The two Solace message routers, along with a SAN attached disk, form a redundant pair.

3.2.1 Persistence

Solace's patented approach to the non-volatile storage of messages called the Assured Delivery Blade (ADB) uses a combination of Field Programmable Gate Arrays (FPGAs), DRAM and proprietary hardware.



When a message is received from a publisher, the Solace message router looks up the destinations and transfers the message payload along with the destinations to the DRAM on the ADB using Direct Memory Access (DMA) and without any system calls or other operating system interaction. DRAM is much faster than even flash memory enabling this transfer to happen with very low latency. Data is streamed continuously to the ADB, instead of with bursty file accesses written from a software thread, which yields high performance and avoids wasting bus bandwidth

While the data is being transferred to the DRAM, the FPGA on the ADB is also transferring a copy of the data (message payload and destinations) to the standby message router via a proprietary mate link with no software interactions or CPU intervention on either system. Both transfers are fully synchronous and the acknowledgement is not sent to the publisher until the message is safely stored in both the active and standby message router.

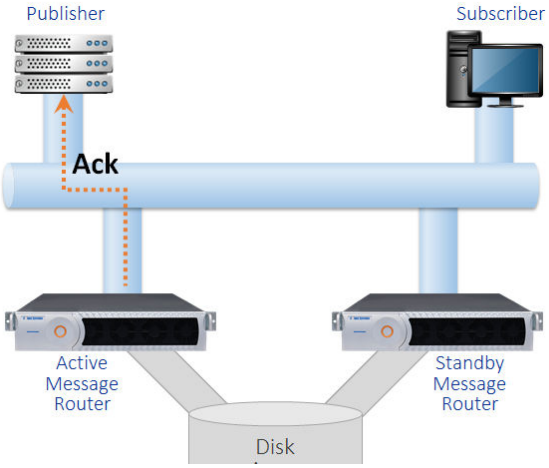
If the standby message router is offline the data stored in the ADB on the active is still 100% safe; data stored in the DRAM of the ADB is non-volatile thanks to proprietary Solace hardware.

Unlike the "one size fits all" nature of block-based storage, the non-volatile RAM on the ADB is byte writeable allowing for efficient data storage for each type of information: large buffers for payload storage, and random access for efficient storage of message delivery state.

3.2.2 Acknowledgement

Once safely stored in both ADBs (active and standby) the acknowledgement can be sent to the publisher. The latency from publisher sending a message, to Solace message router returning an acknowledgement is minimized by the use of DRAM as storage (vs. disk or flash) and by eliminating interactions with the operating system. This roundtrip time is a critical metric because it determines the rate at which a publisher can send messages.

Low latency from message send to acknowledgement translates to a high single thread publish rate, resulting in simpler applications.

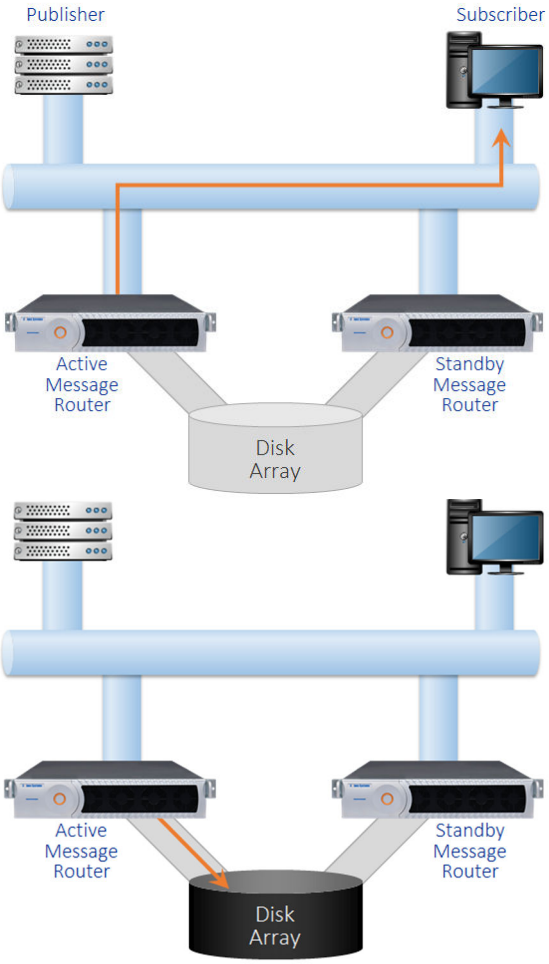


3.2.3 Delivery

Messages are delivered to subscribers from a cached copy in RAM. Only messages that are destined to misbehaving subscribers (offline, previously offline or slow) are retrieved from disk; and deliveries to these misbehaving subscribers are even handled by a separate thread. Deliveries to well behaved subscribers are unaffected by the presence of offline, recovering or slow subscribers. Benchmark test results of Solace products under this type of workload are presented in a separate whitepaper "Advantages and Performance of Solace's Guaranteed Messaging Solution".

Messages that exist for too long in the ADB are moved to mass storage by a background task to recover space; this happens without any interaction with real-time processing tasks. Similarly, deliveries of messages that must be retrieved from mass storage (messages destined for slow or recovering subscribers) are handled separately from newly published messages being delivered to well-behaving consumers.

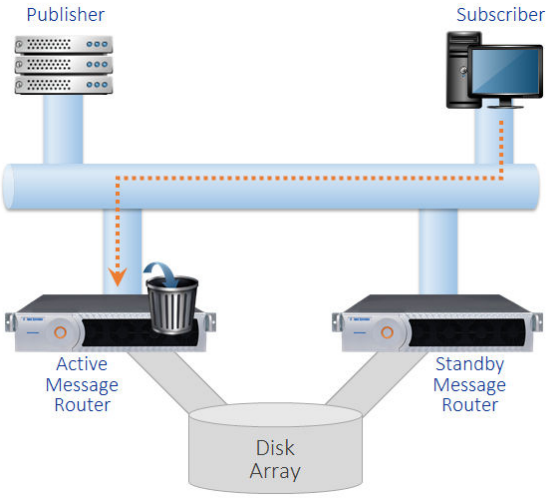
Messages that must be moved from the ADB to mass storage are bundled and written to disk in large chunks making the most efficient use of the disk. There is no latency penalty for bundling the writes since the publisher received an acknowledgment as soon as the message was written to the ADB.



3.2.4 Dequeueing

The Solace message router waits to receive acknowledgements from subscribers and updates the delivery status of the message when it does. Delivery status is stored in the ADB and processing an acknowledgement is a simple DMA operation, instead of a high latency read-modify-write of a file, or a write to a ledger file that must be cleaned upon an activity switch. The update is also mirrored to the ADB in the standby message message router.

Delivery state is maintained in both active and standby message routers resulting in faster failovers and no duplicate transmissions.



4 Enterprise Guaranteed Messaging Requirements

4.1 Performance

Can the middleware scale to keep up with ever increasing data rates and what is the cost and architecture of this scalability?

Solace products offer industry leading performance which allows developers to focus on the business requirements of their applications rather than how to divide message flows across multiple software message brokers. A high performance messaging infrastructure allows for simpler system architectures and simpler applications, which increases developer productivity and reduces time to market.

Software message brokers can only scale by adding brokers in parallel. There is added complexity associated with splitting the message flows between multiple brokers and the scaling is sub-linear since there are always messages that must be sent to clients of both brokers.

Even if the application does not require performance in excess of what a single software broker can provide, savings can still be realized by switching to Solace since it supports virtualization and multi-tenancy. Each Solace message router can support multiple virtual message brokers so a single message router can host many lower performance applications. This typically results in cost savings and performance gains based on statistical multiplexing of applications (assuming application bursts are not highly correlated). Since Solace offers several versions of its internal cards, clients can increase messaging performance and capacity by upgrading to higher performance hardware –a much simpler and more cost-effective way to scale compared to horizontally scaling software brokers and servers.

4.2 Client Isolation

It is increasingly common in enterprise applications to perform real-time or near real-time analytics on a portion of incoming events but to log all events for regulatory requirements or post processing. In many cases the logging application cannot keep up with peak rates; the slow subscriber behavior of the logging application cannot be allowed to affect the level of service provided by the messaging system to the real-time applications.

In fact, as a messaging system scales in terms of clients, so does the importance of per-client isolation since the system is more likely to have slow consumers and if they impact the performance of the system, it is scalability you cannot use. In traditional middleware systems messages for slow subscribers must be retrieved from disk; the extra load on the disk affects the rate at which other clients can publish.

Solace's message routers are engineered to confine the effects of slow subscribers to the clients that are misbehaving. Incoming messages are stored in a high speed DRAM based non-volatile cache. Patented techniques are used to ensure well behaved clients see no degradation of performance resulting from the presence of slow subscribers.

4.3 Redundancy and High Availability

The 24/7 nature of today's global enterprise does not allow for down time. Applications are expected to recover from failure in seconds. A messaging system can have redundancy but, if the switch over time is too long and variable does it satisfy your high availability requirements? Traditional middleware products keep all of their state on disk; if there is a system failure, the standby must retrieve all state from disk before it can resume providing service, which can take minutes or 10's of minutes depending on the amount of data stored. Solace's message routers maintain hot state in RAM on both active and standby systems. Activity switches complete in single digit seconds rather than minutes.

4.4 Disaster Recovery

Critical applications may be required to restart in a secondary location in response to a widespread failure (in many cases, this is a regulatory requirement). Can the middleware support disaster recovery and perhaps more importantly how does enabling it affect performance? Traditional middleware systems use storage based replication to copy state stored on disk to a remote location. Enabling replication on storage equipment further degrades performance. Solace's message routers use patented technology to replicate messages (rather than storage blocks) and transfer them directly without using storage replication. This has many benefits in terms of performance and operational simplicity. Message replication may be done synchronously or asynchronously depending on the requirements of the application.

4.5 WAN Distribution

The global nature of today's enterprise requires large amounts of data to be synchronized across multiple sites. Solace message routers can be networked or federated together. Fanout to local clients is performed by a co-located message router. Solace transfers messages across the WAN using features found in WAN optimization equipment, such as TCP optimizations for long RTT, lossy networks and hardware-based compression. Solace only transfers one copy of each message over the WAN no matter how many subscribers need it, and performs fanout at the remote site. This optimizes bandwidth utilization and maximizes application performance.

4.6 Scalability

With traditional middleware systems the performance of the storage determines message broker performance. Message throughput is decoupled from storage performance with Solace message routers, allowing scalability beyond what is possible with systems that rely on disk storage for message persistence. With Solace products messages are persisted to a non-volatile cache implemented with high performance DRAM until delivered. Only messages that cannot be delivered due to slow or offline consumers are eventually transferred to disk and this can be done in the background so that real-time performance is unaffected. Solace offers multiple versions of ADB and NAB blades at different levels of performance allowing vertical scalability of performance by upgrading existing systems with faster blades. Software brokers can only scale by adding more systems in parallel increasing complexity and possibly requiring application changes to make use of multiple brokers.

4.7 Management

Is the middleware system easy to manage and provide detailed information without impacting performance? What tools are provided to isolate and troubleshoot problems?

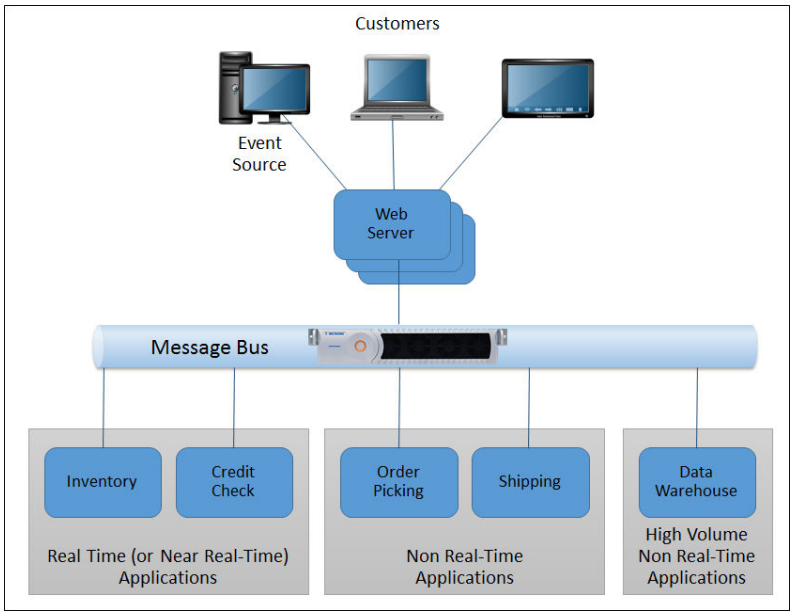
Solace provides a centralized management console that provides real-time monitoring, alerting, graphing and troubleshooting; rich visibility, right down to individual queue and connection statistics without impacting performance and with a level of granularity not available in software message brokers that require operating systems in their data path. In order to meet the requirements of the enterprise, management capabilities must scale with performance. As systems grow in scale or provide a mission critical service, it is imperative that network managers have the tools needed to isolate and troubleshoot problems quickly.

4.8 Total Cost of Ownership

Cost is always a concern. When comparing solutions you should consider CapEx, operating system licenses, software licenses, power, cooling, rack space, etc. Industry leading throughput combined with virtualization and support for multi-tenancy allows multiple software message brokers and their server infrastructure to be consolidated into a single Solace message router. Fewer servers, network ports, savings on software licences, reduced power consumption and simplified management add up to cost savings.

5 Case Study – Online Retailer

The diagram to the right shows an example enterprise application making use of guaranteed messaging. The example is an online retailer, but other applications such as a telecommunications service provider portal or online broker are similar in structure.



There is a source of events, in this case customers clicking on items they wish to purchase. These events are gathered up by web servers that communicate with the messaging system on the back end. Events such as a click to place an item in the customer's basket are interactive and must be handled in near real-time.

For example, before the item shows up in the customer's basket the web server must make a request to the inventory management system to see if the item is in the warehouse and reserve it. Similarly if the customer proceeds to check out and complete the transaction, the system will need to check the customer's credit card details to make sure the card is valid and has sufficient available credit. Studies have shown that delays in processing these interactive events result in frustrated customers that complete fewer transactions.

The real-time applications must be provisioned to run at peak load. From the point of view of the guaranteed messaging system, these applications are well behaved since under normal circumstances they do not allow messages to accumulate in queue. Once the transaction is completed another series of events is generated where the order is dispatched to the warehouse to be picked, packaged and shipped.

In the case of the online retailer it is ultimately humans responding to these events, as such they are not real-time in nature. For these non-real-time applications it does not make sense to incur extra cost to provision these systems to be able to handle peak load and the applications may regularly use the guaranteed messaging system to queue events, for example until someone is available to pick the order.

There is a third class of application and that is the data warehouse which for reasons legal or otherwise must record every event. Depending on the application it may not be practical to provision the data warehouse to keep up with peak rates. Under peak loading the data warehouse application is expected to get behind and use the guaranteed messaging system as a shock absorber queuing events until a time when system loading is light and the data warehouse application has an opportunity to catch up.

This example illustrates a few key points. First, there is a mixture of applications which must respond in real-time and applications which are not required to respond in real-time. Second, even under normal circumstances some applications may not be able to keep up with peak rates and require the guaranteed messaging system to queue messages.

The guaranteed messaging system must, under this type of workload, be able to provide sufficient performance to be able to keep up with new event arrivals, providing real-time performance to applications that require it while queuing messages to relatively slow warehousing applications that are not designed to be able to keep up at peak loading. The messaging system must be able to provide real-time performance to applications that require it while queuing messages to slow subscribers and it is most likely to be required to operate in this mode under peak loading. In the example online retailer application, failure to provide this type of performance will result in lost revenue.

6 Summary

Solace's guaranteed messaging system overcomes the limitations of disk-based storage to provide better performance and robustness than software-based solutions under the conditions typically found in modern application architectures.

A patented hardware-based approach streamlines the process of queuing, acknowledging, delivering and dequeuing messages to offer higher throughput, lower latency and better isolation of clients so the backlogging and recovery periods that disconnected and slow consumers introduce don't affect the system as a whole. Solace message routers also reduce the cost and complexity of guaranteed messaging systems by handling a volume of messages that would require dozens of software brokers running on dedicated servers. Finally, Solace improves system reliability with redundant components, automatic failover, and even fully-integrated disaster recovery facilitated by synchronous real-time replication to backup systems across WAN links.