Real-Time Middleware Seminar 2007

April 15, 2007

ProMo - A Scalable and Efficient Middleware for Real-Time Online Data Delivery

Haggai Roitman, Avigdor Gal (*Technion – Israel Institute of Technology*), Louiqa Raschid (*University of Maryland, College Park*)

Applications that require online data delivery include RSS feeds, stock prices and auctions on the commercial Internet, context aware services in the Mobile arena, and increasingly, the availability of Grid computational resources. Services for clients in such a setting have become increasingly sophisticated and customized, in order to meet a diversity of clients and their profiles. This has challenged the capabilities of both service providers and current data delivery middleware. To satisfy client needs, data delivery middlewares use either push or pull methods and each suffers some drawbacks. Several hybrid push-pull solutions exist (e.g., [1]), which rely on servers to predict when a client is about to monitor. This does not address the scalability problem.

To illustrate the problem, consider a simplified scenario of a physicist client, interested in executing a CPU and memory intensive simulation on a Linux Farm consisting of several thousand nodes. The Linux Farm collects Node Status metadata for each node including attributes such as Machine Up Time, CPU Load, Memory Load, Disk Load, and Network Load; this data is updated every time a new job is submitted to the Linux Farm and is pushed to interested clients every 10 minutes. She further determines that a remote dataset is needed and it can be retrieved from a number of replica sites. Network Behavior metadata for these replica sites are available from either a GridFTP monitor (updated with each new request to the replica site) or the Network Weather Service (NWS) monitor (updated every 5 minutes). Neither monitor pushes updates to clients.

We present ProMo, a generic middleware that includes a language, a model, and algorithms to support flexible, efficient and scalable real-time online data delivery. It is flexible since it can accommodate push, pull or hybrid push-pull solutions. It is efficient since it exploits server capabilities in meeting client needs. It is scalable since it exploits aggregation of client profiles, and efficiently distributes the server workload, based on server capabilities. ProMo provides a uniform specification language that can be used by clients to specify client data needs and by servers to specify server data delivery capabilities. ProMo is event-based, using the abstract model of execution intervals, presented in [2], that capture periods of time where client needs should be met or server capabilities can be utilized to construct an e¢ client push-pull schedule.

Our proposed middleware reason with execution intervals to generate efficient schedules of task monitoring. We also propose an adaptive approach, in which ProMo utilizes information from the server to update its execution intervals and incrementally generate new schedules

online. ProMo can manage multiple clients, providing heuristics that offer a trade-off between the amount of captured events and the delay in which they are provided to clients.

References

[1] P. Deolasee, A. Katkar, P. Panchbudhe, K. Ramamritham, and P. Shenoy. Adaptive push-pull: Disseminating dynamic web data. In Proceedings of the International World Wide Web Conference (WWW), 2001.

[2] H. Roitman, A. Gal, L. Bright, and L. Raschid. A dual framework and algorithms for targeted data delivery. Technical report, University of Maryland, College Park, 2005. Available from http://hdl.handle.net/1903/3012.

Event-Driven Approach for Real-Time Enterprise

David Botzer, *Manager, SW Tools & Event Systems, IBM*

Gartner's definition of the Real-Time Enterprise: "The RTE is an enterprise that competes by using up-to-date information to progressively remove delays to the management and execution of its critical business processes."

Concepts such as: RTA – Real-Time Analytics, ZLE – Zero Latency Enterprise, STP – Straight Through Processing and RTE – Real-Time Enterprise are commonly discussed.

However, if you ask what are the expectations or definition of "real-time" is in the business, you will not get a clear and consistent answer. Certainly, we do not want to see histograms of expenses and revenues bouncing up and down in real-time. We don't want senior management making real-time decisions, so they can make even more decisions every day. So where is "real-time" delivering value today?

The most analyzed and reported value of "real-time" is in improving the efficiency of operational systems. Examples of these successes are in the areas of: Fraud Detection, Electronic Trading, Call Center, Operational "Control Panel". This type of "Real-Time" technology has been focused on transactions and how these events can be handled more efficiently. These recent technological advances and their associated products have addressed "speeds and feeds" issues. These products are capable of monitoring hundreds of thousands of transactions per second and apply complex rules to these transactions in real-time. Most likely, every time you use your credit card, a real-time system analyzes it for possible fraudulent use.

The role with the real-time monitoring of operational systems has been to provide context via the Business Integration Platform. For example, it is not particularly useful to simply alert a manager to an event that has exceeded some threshold. It is far more useful to alert the manager to a specific event within the context of other related information and relevant history. Now, the managers have more information at their fingertips and the ability to analyze that information. For example, IBM has been used in conjunction with IBM Active Middleware Technology at a major financial services enterprise to greatly improve the efficiency of the detection and reaction to Events.

Real-Time Messaging: Financial Industry, and Beyond

Gidon Gershinsky, *Messaging Architect, IBM*

The exponential growth of data volumes in financial front office markets (stock exchanges and institutional traders) does not show signs of relenting, with typical throughput rates already reaching hundreds of thousands of messages per second, and projected to surpass a million messages/sec in just a year or two. In addition, the end-to-end data delivery latency is required to stay well below a millisecond threshold, due to the increasing popularity of program (automatic) trading. The stringent limits are put not only on the average latency, but also on the worst-case delivery time. This type of performance requirements calls for special-purpose real-time solutions.

Other industries (such as defense, telecommunication, industry automation, gaming, etc) do not yet reach the data flow volumes of financial front office, and therefore consume a more traditional middleware technology. However, the performance requirements - throughput, latency, and predictable quality of service, are becoming increasingly important in these industries as well - leading to the necessity of applying real-time methods to the traditional large-scale middleware solutions.

Our team works on research and development in both financial front-office and traditional industry domains, with technologies ranging from special-purpose financial to large-scale generic real-time messaging. In this talk, I will describe the challenges and solutions in this space.

The Tendency of the Embedded Market

Arik Weinstein, *Senior Application Engineer, Mentor Graphics Israel – Embedded Division*

Today's breakneck development schedules leave little time to write code that doesn't show off your team's distinctive competence. To make the most of your time, manage complexity and deliver with confidence, a high quality software platform has become a necessity.

The Nucleus OS is a proven embedded operating system that works out of the box and lets you focus your efforts on differentiating your products.

- Source code provided
- Royalty free!

- **Nucleus Kernel**
  Real time kernel, C++ and POSIX interfaces, dynamic download, interprocessor communications
- **Nucleus Networking**
  There are many ways for devices to communicate. The Nucleus OS provides over 60 network drivers and protocols including TCP/IP stack, IPv6 and 802.11 wireless to provide exactly what your product needs.
- **Nucleus GUI**
  Ease of use for the consumer starts in the graphical display. The Nucleus OS provides a wide range of services to help develop a powerful and intuitive GUI.
- **Nucleus File System**
  Storing and retrieving data is prevalent in today's devices. The Nucleus OS helps manage this through its file system software.
- **Nucleus USB**
  Consumers demand an easy interface for connecting devices to each other. The Nucleus OS supports the USB standard for this request. The vast number of class drivers supported allows for versatility in your application.
- **Nucleus Bus Support**
  Communication across busses is becoming more and more commonplace. The Nucleus OS provides support for PCI, PCI-X, CAN, SPI, and I2C.
- **Nucleus Security**
  As the world relies more and more on digital storage and communication, the risk of private data becoming public increases. The Nucleus OS provides an assortment of encryption and hashing technology to help overcome this risk.

Second generation of event processing

Opher Etzion, *Senior Technical Staff Member, Lead Architect, Event Processing ESB Technical Strategy, IBM Software Group*

The first generation of event processing products, currently pervasive in the market, is centered on the notion of monolithic event processing engines that execute collection of rules, scripts or queries, depending on the engine's API orientation. The first generation products have challenges in several areas, such as: coping with complexity, supporting high-throughput, scalability in a massively distributed environment, and providing different quality of service in processing, supporting real-time constraints. Analysts now indicate that a second generation of event processing is emerging. Some vendors already reveal some of the characteristics of the second generation.

The talk will start by surveying existing architectures, used by current vendors, and point out the challenges that the first generation solution is facing. It will then concentrate on the principles of the second generation, from architecture point of view, move from monolithic engines to agent architecture, in which each agent is lightweight, carries a single function, and may have individual quality of service properties. The architecture and all its components will be described. Some more of the issues that will be discussed are:

1. Layered architecture: locating different functions in different layers, enabling massively distributed system.
2. Exploiting parallel architectures: parallelism inside specific layers that are implemented in parallel hardware architectures.
3. Language issues: towards semantic view of event processing
4. Role of standards: the different facets of interoperability
5. Support of real-time constraints.

Dynamic Real Time Complex Event Processing

Greg Porpora, *Real-Time and Complex Event Processing (CEP), IBM Federal*

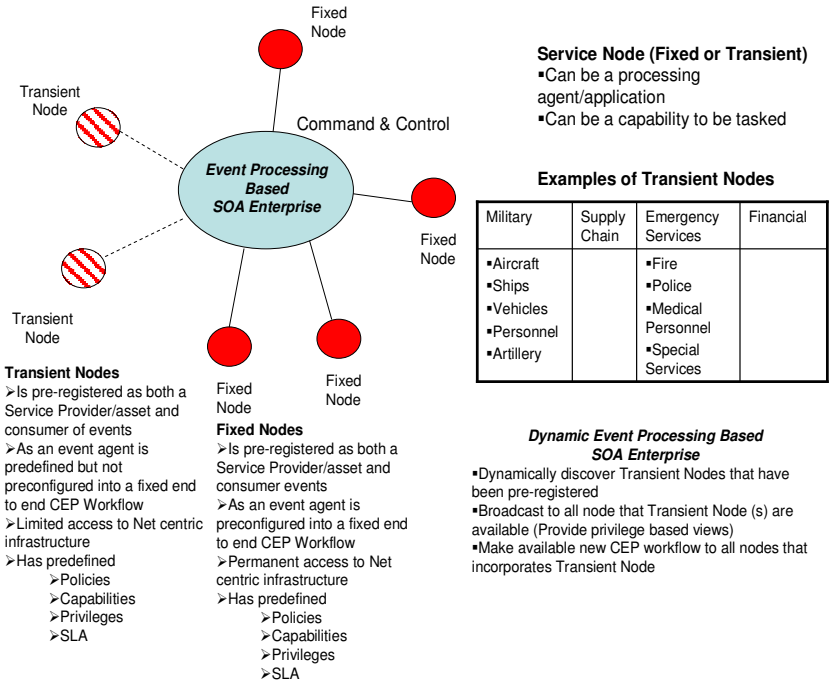
This paper will address the challenges, requirements and application of dynamic Complex Event Processing (CEP) enterprise within the context of a military as well as emergency services scenario. This class of RT CEP is targeted towards the third generation of event processing systems.

The first generation of event driven systems was based on predefined events and node to node relationships that performed relatively simple event pattern detections coupled with various transformation capabilities ranging from aggregation, translation, composing to splitting. Unlike Web services, the Internet or recently Service Oriented Architecture (SOA), none of these environments were based on widely adopted standards or reference architecture and each was a unique solution set of technologies targeted to a specific problem domain or customer. These first generation systems had fixed end to end Event Processing (Complex or Simple) workflows and Service Level Agreements (SLA's) that were predefined as part of their respective registration and system development. There was no ability to dynamically discover albeit were pre-registered nodes/ that exhibited transient availability or access. This capability would require an event process system to dynamically modify the end to end event process workflow and impacts to the overall SLA to event consumers. Event consumers and their subsequent actions were limited to this fixed environment and could only be modified via a development effort and follow-on redeployment to the production system. This first generation set of technologies and systems were a great learning environment to examine the temporal and causal constraints and challenges of performing event processing across an enterprise of heterogeneous infrastructure. The advent of standards based Net Centric Computing has been a key enabling technology for capabilities such as Global Supply Chain, On Demand Computing, Net Centric Operations and Warfare (NCOW), Global Real Time Financial trading, etc. The convergence of technologies has pushed the event processing community realize that the second generation of event processing technologies must be based on a set of industry defined and adopted standards and definitions.

Work is on-going now to define what form the second generation of event processing systems will be based on. Organizations such as the OMG and the Event processing Consortium are aggressively working to define a common set of event definitions and terminology, Event and messaging data formats, a set of common reference architectures targeted to specific industry ontology's and developing a converged model for stream-based and event-based processing approaches. All these on-going efforts are expected to converge in the 2007-2008 time-frame with a determination as to how SOA and Event Process interrelate.

However, the second generation of event process systems will still be static albeit more capable and standards based. The ability to dynamically discover, integrate and enable event producers and consumers that are not fixed is a specific infrastructure/enterprise will need to be addressed in the third generation of event processing systems.

One would ask why is this important?, what type of transient event producers or consumers would be applicable to a fixed pre-defined Complex Event processing workflow and is the added complexity, intelligence and development required to enable an event based architecture or Enterprise Service Bus (ESB) off set by the market opportunity.



In reality transient based event producers and consumers are part of the every day fabric of many types of human and machine interactions. These unscheduled interrupts or timely discovered capabilities/assets are constantly being exploited to address real time or just in time needs that are critical for a successful outcome that has been subject to unforeseen external actions. As such, systems will need the ability to be reconfigured to dynamically scale up or down based on the persistence of transient assets as well as have user interface/tooling that allow for manual integration of non pre-registered assets:

Pre-registration of transient resources/assets as part of base development effort which include the following attributes:

- Events QoS processing with the Transient node
- Interface requirements/protocols
- Assignment to a predefined Community of Interest (COI)
- Capability (process, transform, route, enrich)
- Relationship to other fixed or transient assets in COI (Inputs, Outputs)
- Discovery when asset interrogates network with proper authentication, access rights and security level
- Dynamic assignment or deletion to a COI's
- Dynamic reconfiguration of node-to-node or end-to-end Complex Event Processing flow QoS with special attention to alternate rules set within Event processing agents

Manual Registration of undefined assets:

- Authentication, Access privileges
- Interface/protocol definition
- Assignment to COI and relationship
- Capability

Real-time scheduling algorithms for fault diagnostics of computer communication networks

Eugene Levner (*Department of Computer Science, Holon Institute of Technology*), David Alcalde (*University of La Laguna, Tenerife, Spain*), Vladimir Vishnevsky (*Institute of Information Transmission Problems, Russian Academy of Sciences*), Sergey Frenkel and Viktor Zakharov (*Institute of Informatics, Russian Academy of Sciences*)

We consider a problem of the fault diagnostics in computer communication networks aimed to minimize the expected losses caused by the faults. Given a chance to overlook a fault during the testing, a subsystem (module, component) can be inspected for a fault more than once. We develop a fast real-time asymptotically-optimal algorithm that yields an optimal sequence of the inspection steps. Optimality conditions under which the (local) real-time scheduling procedure yields the global optimum are found.

Our approach to the fault diagnostics is based on using test transactions, or *probe*s. A *probe* is a test transaction whose outcome depends on the status of some of the system's components; the diagnosis is performed by appropriately selecting the probes and analyzing the results. For example, a probe can be a program that is executed on a particular machine called a *probe station* by sending a command or transaction to a network component (say, a server or other network elements) and measuring the corresponding response. The well-known program *ping* is a common example of such a probing tool that can be used to detect the network availability. Also, the probes can be sent in the form of test e-mail messages, web-access requests, and so on. In the context of the probing paradigm, a system usually is represented by the "*dependency graph*" supplied by a "*dependency matrix*" where nodes can be either hardware elements (e.g., workstations, servers, routers), or software components, or services, and links can represent both physical, technological, and logical connections between the elements.

To get the information presenting the dependency matrix, it is necessary to simulate and to prototype the system model, which is time-consuming and costly. At the same time, some early estimation of the system characteristics to be diagnosed could be obtained by studying faulty modules by various optimal search theory strategies (Ahlswede 1987, Levner 1994, Alcaide et al. 2006), when the search effort resources are limited and only probabilities of the possible location and possible error of fault detection of the modules are given.

We model a computer system as a collection of known components interacting with each other. There are n independent, stochastically failing modules. When the failure occurs, a series of sequential inspections is performed in order to identify the failed module. The goal of the search is to minimize expected losses incurred by the system failure before the latter is detected.

The model involves the following parameters: p_j, a prior probability that j-th module is faulted, q_j, probability of successfully detecting the fault in module j, ρ_j, the probability that inspection of the node j can fail when it is fault-free, τ_j, the expected time to inspect module j, l_j, the expected loss rate per unit time, respect to module j, L, the max possible loss level.

Main results. (1) Consider the Bayesian min-loss fault diagnostics problem with the linear and exponential cost criteria. Let the values of the *search effectiveness ratio* R_i be $C_i t_i(1-$

$p_i q_i)/q_i$, for all values i. Let R_i be arranged in non-increasing order of magnitude, and let the search sequence s^* be such that if the ratio R_i is the nth largest value in the ordering then the n-th step in s^* is the next search of component i. Then the sequence s^* is optimal. (2) Linear and exponential functions are the only search objective functions in the Bayesian search case for which a local real-time search procedure guarantees a global optimum.

Wind River's Hard Real-Time Solutions

Moti Elkayam, *Technical Account Manager, Wind River*

Wind River Real-Time Core for Linux enables hard real-time response for applications ranging from single-core feature phones and high-bandwidth IP communications to robotics and industrial control. The industry considers this technology one of the best, most mature hard real-time Linux solutions available today. Real-Time Core includes a hard real-time executive that coexists with the Wind River Linux kernel. Together, they combine the hard real-time capabilities necessary for high-performance deterministic responsiveness with an optimized, integrated, commercial-grade Wind River Linux platform.

Optimized for Developers

Wind River Real-Time Core and Linux provide device manufacturers with mature, proven technology for developing complex, next-generation Linux-based applications requiring hard real-time with microsecond-level interrupt and scheduling latency. Real-Time Core employs a simple real-time executive that runs the non-real-time Linux kernel as its lowest priority task and routes interrupts to the Linux kernel through a virtual interrupt layer. Real-Time Core initially handles all interrupts and passes to Wind River Linux only when there are no real-time tasks to run. Real-time applications are loaded in kernel space and receive interrupts immediately, resulting in near hardware-threshold speeds for interrupt processing. Wind River Real-Time Core and Linux user tasks communicate through lock-free queues and shared memory in the current system. From the application programmer's point of view, the queues look like standard Linux character devices, accessed via POSIX read/write/open/ioctl system calls. Shared memory is currently accessed via the POSIX mmap calls. In this scenario, the Linux kernel is largely untouched. However, the interrupt abstraction model does have some differences from "standard" Linux in that the real-time tasks must be implemented as Linux kernel loadable modules. This is the same format used for other kernel features, such as file systems and device drivers.

Features and Benefits

Real-Time Core provides hard real-time capabilities combined with easy development, integrated tools, and rapid deployment of Wind River Linux. Real-Time Core is both standards-based and General Public License (GPL) safe to protect proprietary and IP assets.

Hard real-time application determinism performance is guaranteed to the threshold limits of the underlying hardware, with microsecond response times across a broad range of architectures. Real-Time Core allows you to leverage the optimized integration, knowledge base, and commercial-grade services and support of Wind River Linux, plus the open-source community for drivers and applications. Simple POSIX interfaces allow hard real-time applications to be nearly indistinguishable from standard UNIX applications. This helps you leverage existing skills to accelerate development time and deliver products to the market faster.

CEP for Location Awareness and Embedded Applications

Ella Rabinovich, *IBM*

The future of ubiquitous computing promises a world in which we are surrounded by smart devices that maintain current information about their locations, the contexts in which they are being used, and relevant data about their users. As wireless data communication technology becomes more widespread, and as embedded computers become less expensive and more powerful, it will be increasingly possible to gather extensive information from heterogeneous data sources. This presents novel opportunities for enhanced services that can make timely use of this data in a wide range of application domains. However, extracting meaningful and actionable information from these raw streams of data in real time remains a challenge. Applications will require flexible and scalable mechanisms for constructing condensed, refined views of the data, possibly in ways unanticipated by the data providers. Complex Event Processing (CEP) is an effective solution to this problem. CEP correlates the data (viewed as events streams) in order to detect and report meaningful predefined patterns, thus supplying the application with an effective view of the accumulated incoming data (events). CEP engines provide a flexible mechanism with which to define and modify these patterns (rules), thus providing a flexible mechanism for applications to construct the desired condensed and refined views of the data. In order to distribute processing responsibility, reduce unnecessary network traffic, improve response time and provide the ability to operate in disconnected mode, it is desirable to be able to perform CEP directly on an embedded device. In response to these needs, we have developed a CEP engine for the J2ME environment. This paper describes the architecture of the embedded CEP engine, and the challenges that had to be solved in order to implement it on a relatively resource constrained device. We motivate the value of our approach by showing how the embedded CEP paradigm effectively supports the requirements of a mobile patient care scenario involving medical monitoring and alerts.

Delay Analysis of Real-Time Data Dissemination

Harel Paz, *IBM*

The growing popularity of distributed real-time applications increases the demand for QoS-aware messaging systems. In the absence of transmission rate control, congestion may prevent a messaging system from meeting its timeliness requirements. In this paper, we develop an analytic model for congestion in data dissemination protocols and investigate the effect of transmission rate on message delivery latency. Unlike previous works, we take into account the processing overhead of receiver buffer overflow, which has a significant impact on the results. A simulation is used to obtain more insight into the problem and to study a number of additional effects ignored by the analytic model. The presented analysis can be incorporated in a transmission rate control logic to estimate the expected latency for a particular transmission rate.

Real-Time Programming in Java

Konstantin Shagin, *IBM*

Java is not the first platform that comes to mind when planning to write a real-time application. Real-Time Specification for Java (RTSJ) aims to change this perception by standardizing a set of APIs and JVM semantics which allow Java developers to control the temporal behavior of their applications. This talk gives an overview of RTSJ basic principles and most notable features.