

Murky Transparencies: Clarity using Performance Engineering

Joseph Martinka, Richard Friedrich, Tracy Sienknecht

Hewlett-Packard Company - HP Laboratories, Palo Alto, California U.S.A.
{martinka, richf, tracy}@hpl.hp.com

This position paper highlights a daunting challenge facing the deployment of open distributed applications: the performance management component of the transparency functions. Applications operating in an ODP environment require distribution transparencies possessing comprehensive performance management capabilities including monitoring and modeling. The transparency functions are controlled by adaptive management agents that react dynamically to meet client QoS requirements given a current set of server and channel QoS capabilities. This technical challenge must work in an open environment with multiple autonomous administrative domains. For this goal to be realized, the ODP architecture must be enhanced. Distributed performance management of “operational” communications has been neglected in favor of the trendy multi-media “streams” communication in spite of the dominance of the former in current and future applications.

Keyword Codes: C.4, I.6.3, C.2.4

Keywords: Performance of systems, Simulation, Distributed systems

1 ODP’S CHALLENGE TO APPLICATION PERFORMANCE

The ultimate goal for applications in Open Distributed Processing (ODP) is to insulate the application design and programming from the effects of distribution [1]. Such a goal is at once noble and daunting. The design space for even simple distributed applications using current API’s such as the Distributed Computing Environment (DCE), Corba, Banyan VINES, or other systems, discourages the average application designer. The ODP goal to insulate the application from its distributional complexities decreases the time, risk and expertise needed to design workable ODP applications. However, the inevitable consequence of making the application transparent to distribution is that the infrastructure must assume the role of providing the resulting transparency mechanisms. As Hamlet reflects: *ay, there’s the rub*. The architectural specification for the mechanisms that support the application in meeting its Quality of Service (QoS) and functional goals while providing transparent distribution missing from the Reference Model for ODP (RM-ODP).

1.1 The changing application environment

Meeting this requirement for distributional transparency will be difficult enough even if simply limited to a single distributed systems infrastructure (e.g., DCE) which permits heterogeneous hardware. The future ODP infrastructure is heterogeneous in software and hardware. This fact compounds the performance management challenges. Systems will be composed of multiple vendor operating environments running various distribution software (e.g., DCE, CORBA, SNMP, TINA-based DPE) that must interoperate. The communication channels will deliver concurrent asynchronous packets for operational processes and isosynchronous stream traffic with radically different control and QoS criteria.

Since the ODP architecture is the only unifying core to unite this Babel of systems, it is insufficient. Few programs will be written without substantial assistance and reliance on

middleware services and libraries. The use of middleware will be composed of many “black box” functions that are harder to characterize and tune since they are not directly accessible by the same tools developers use for their own code. The application itself becomes more obtuse. Having parts of a client /server application written by the same developers is likely to become the exception than the norm. Applications will increase their performance dependence on services not under the control of the application designer. Implications of the functional choices available to the developer are often hidden and may not match the assumptions of the middleware design. In some cases, these middleware services will be based on legacy application ‘back-ends’ expected to exist in an ODP environment for many years. As the distribution infrastructures mature, some applications will outlive the infrastructure on which they were built. Optimizations appropriate for one environment can be sub-optimal for its replacement. Design trade-offs for future infrastructures are impossible to predict or anticipate. Some means to cope with this change must be provided during the application’s lifetime.

Networking channels introduce variable and uncertain latency factors into an application’s performance not faced in conventional monolithic applications. Network distance between a client and server is not controllable at design time, and may range from a co-located process in the same node to a node on another continent. The result is wide deviations of average latencies of communication, compounded by variability due to network congestion effects. These latencies are troublesome even as bandwidth improves markedly in the future.

The performance challenge in ODP should be met by applying both existing performance engineering techniques and those requiring breakthrough technologies (e.g., automated performance management control). The breakthroughs need an aggressive research and development effort and architectural support from the RM-ODP if the goals of ODP are to be realized.

1.2 Performance Issues with Transparencies and Domains

Most ODP transparencies have significant performance components, particularly *location*, *migration* and *replication*. Each needs to provide performance management functions that will maximize the performance behavior of the environment.

In *location* transparency, a server object instance is found for the client object. The selection of this server among several choices should be made based upon performance factors, including which server is best-suited to meet the client’s QoS needs. These factors should be based in part on the channel latencies and contention between the client and a proposed server object, current loading of participating nodes, etc. We believe that to satisfy this transparency, automatic agents must manage the location transparency (trading and binding) mechanisms.

In *migration* transparency, an object is moved from one node to another in order to maximize the ability of the server object to meet a larger fraction of the client binding requests with satisfactory QoS. This transparency enhances the system’s ability to balance the load, reduce latency, move object servers to follow periodic or unexpected workload patterns, and to unload hardware nodes for administrative or maintenance work.

In *replication* or group transparency, the use of mutually behaviorally compatible objects act together to support an interface and enhance *performability*: the performance and availability of a service. Replication carries a performance cost. As it is utilized, the overhead to maintain consistency between servers increases the resource consumption among replicates. Performance management agents must ensure that the number and geographical placement of replicas achieve the most efficient gain in performability.

An additional requirement for performance management is that automated performance management agents interact in a federation or domain. These agents, for the sake of scalability and complexity, must negotiate and trade information among themselves so that transparencies can function efficiently on behalf of clients. Agent management operation will minimize human interaction to the extent possible to allow greater administrative spans of control, and handle interactive loads several orders of magnitude larger than current day systems. Negotiations with agents outside that federation (e.g., services provided to or for another commercial entity) must be in terms of common QoS agreements, the methods and terms of which are little understood today.

If automation is to be used, only systems that are sufficiently and thoroughly understood can have automatic transparencies. Precise definitions of systems boundaries and adequate abstraction of the world outside these boundaries are areas where RM-ODP must assist.

2 PERFORMANCE MANAGEMENT: CAN IT ASSIST?

The dynamic nature of shared network channels and binding implications on QoS require an automatic element in ODP performance. Decisions to select efficient bindings, migrate objects to improve efficiency and replicate objects for load-balancing require a performance decision-making apparatus which is dynamic and mostly autonomous. Predictable system behavior results only if we architect appropriate performance management functionality, a common performance management control language, and develop objects with expectations to cooperate with the environment.

2.1 Instrumentation and monitoring

Pervasive, heterogeneous, distributed application instrumentation is essential for the performance management of the ODP environment. The management difficulty is increased by using many disparate technologies, distribution protocols and operating environments. We developed consistent performance metric requirements for this instrumentation [2] and developed a specification for distributed monitoring in DCE [3]. These efforts provide insight into the performance management needs of ODP applications such as providing users with a single coherent view of application behavior regardless of application object location. Distributed performance management needs extensions to RM-ODP in two areas: standardized performance metrics and standardized access and control mechanisms.

Pervasive, standard metrics provide the crucial foundation for distributed application performance management. These metrics support evaluating computational, engineering and technology viewpoint behavior in relation to the enterprise and information viewpoint requirements. We define implementation instances of the metrics as sensors. Sensors must be pervasive in object, cluster and capsule software services to realize end-to-end QoS goals. Thus standard metrics allow consistent interpretation of data collected anywhere in the domain.

However, standard metrics are necessary but insufficient. Standard sensor access and control mechanisms are required. These mechanisms should be implemented as a Performance Measurement Interface exported by all objects. An additional mechanism is necessary to support efficient and scalable performance measurement. Sensors must support management by exception semantics through a threshold evaluation technique. Configuring a value and percentile results in sensors that report data only when the threshold conditions are met. There is also a critical need for performance tracing capabilities in ODP systems so as to understand behavior of a workload's transactions. The trace provides logical topology information needed to model the resource use of a transaction as it demands various application services.

A scalable measurement infrastructure must be defined to collect and transport performance data efficiently without noticeable resource utilization. The measurement infrastructure must also support correlating metrics from objects that reside on different computing nodes.

2.2 Performance modeling

The requirements for distributed modeling extends the traditional modeling boundaries concentrating on classic computer node problems. The system boundaries now include the network as a vital component to application throughput and response times. Transitions between abstraction levels are usually painful because current modeling techniques and assumptions for capacity planning are ill-suited for this increased scope. The unit of work, the user transaction, which normalizes the resource demands of the system must be distinguishable and traceable through the layers of middleware as well as opaque services/objects.

The interactions of diverse resource behaviors makes modeling more challenging. We found that our models' outputs depend less on the detailed specification of node's resource use, and more on the understanding of the application's use of communication channels. Model details become entwined with the location and frequency of access to various server objects separated by uncertain network latencies. Consequently this complexity will prompt the model to be delivered with the object. The ODP application modelling will become an operational capability for critical ODP transparencies beyond its traditional role in capacity planning and performance tuning. It is also needed in ODP application design to extend or extrapolate prototype or benchmark measurements to larger target systems.

3 CONCLUSION

Instrumentation and monitoring must be integrated with modelling so that automatic parameterization of models can support the decisions of location, migration and replication transparency agents. Models can be used to assist the synthesis of end-to-end QoS expectations where comprehensive measurement is impossible or burdensome. Proxy instrumentation can occasionally or constantly validate the models as conditions change operationally or as new applications are fitted into the environment. There is substantial interest and research in the issues of multi-media streams QoS. However, the prosaic operational channels also need additional ODP architectural support to ensure that the promise of distributional transparency for performance can be built into supporting agents and managers. Specifications in QoS for operational performance must be enhanced and combined with robust performance models providing "real-time" results. This synergy must be developed to support distributional transparencies. Monitoring, collection, modeling and controlling of the system will need largely be automated to realize performance critical transparencies. These technologies are not yet understood nor pervasive in today's distributed environments. ODP requires them tomorrow.

- 1 ISO/IEC JTC1/SC21/WG7 N885, *Reference Model for Open Distributed Processing Part 1*, November 1993.
- 2 Richard Friedrich, *The Requirements for the Performance Instrumentation of the DCE RPC and CDS Services*, OSF DCE RFC 32.0, June 1993.
- 3 Richard Friedrich, Steve Saunders and Dave Bachmann, *Standardized Performance Instrumentation and Interface Specification for Monitoring DCE Based Applications*, OSF DCE RFC 33.0, November 1994.