eOS – the dawn of the resource economy

John Wilkes, Patrick Goldsack, G. (John) Janakiraman,
Lance Russell, Sharad Singhal, and Andrew Thomas

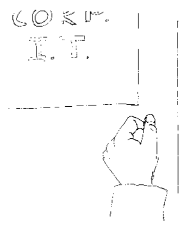
<wilkes@hpl.hp.com>
Hewlett-Packard Laboratories, Palo Alto, CA 94304

21 May 2001

Achieving the benefits of a resource economy*, which supports the execution of services wherever and whenever is most convenient, cost-effective, and trustworthy, represents the next big computer systems research opportunity. That is, we believe that the emphasis in the operating research community should move away from working out how to extract a few more percentage points of speed from individual computing resources, and focus instead on how to size, provision, and manage those resources to serve the needs of a rapidly diversifying set of services.*

HP Laboratories are embarking on a major endeavor to pursue this, and this position paper outlines our view—and a taxonomy—of some of the issues involved, and highlights some of the research opportunities that it presents. We are actively seeking research partners to collaborate with us.

Note: this is a longer form of the position summary that is to appear in the proceedings of HotOS-VIII (Schloss Elmau, Germany, 21-23 May 2001).



Scene: a Corporate IT director's office, the day before a company board meeting. The COO, Chris, knocks on the door, and comes in without waiting.

Chris: Jean: what's all this about us getting billed for computing resources in Singapore? How am I going to explain that? We don't have a facility there! What's going on here?

Jean: Calm down! It's ok—really.

Chris: Not good enough. You know I have to give a bullet-proof answer tomorrow … so you don't have much time.

Jean: Ok, ok. Do you remember the end of the last month? We had the R&D guys needing to do their protein shape calculations to meet an FDA deadline …

Chris: Yes—but weren't they using some cheap compute cycles in Prague that you'd found for them?

Jean: … and then the marketing team wanted a new computer-generated-graphics commercial in time for Comdex that included film of our Malaysian manufacturing plant as a backdrop …

Chris: Yes—but you said that wouldn't be a problem …

Jean: … and the financial results that you are holding, by the look of it, needed some decision analysis that we don't usually do, in collaboration with our Japanese partners …

Chris: Yes—but you told me …

Jean: Please—if I could finish?

Chris: Sorry. It's been a bit hectic today.

Jean: No way could the Prague facility keep both the chemists and the video team happy: both the computation needs and storage space requirements were way over the top. And the Malaysian plant has nothing, really, so it looked like it would cost us a fortune. … All because those geeks couldn't get their timing right.

Chris: [Sigh.] I must have told them a dozen times …

Jean: But then our eOS system discovered that we are co-buying storage space for the Malaysian lot-failure analysis data with our Hong Kong subsidiaries in Singapore; and it checked out the effects of migrating the data back to Prague and the computations to Singapore.

Chris: But wouldn't that have been a huge hassle to get right? Moving all that data?

Jean: Not at all—I didn't even find out until 2 days after it had happened!

Chris: What do you mean? You let it make a decision like that?

Jean: Sure! It even reported that the average response time for our OLTP jobs were 30% better than usual—they usually get hammered by the decision support people at the end of the month. Probably because of the time-zone effects. If you look, I think you'll find we even saved money - we used to have a team of people doing this stuff, trying to keep one step ahead of the next wave of demands. They could never keep up, so the customers were always unhappy—and they were people who we couldn't really afford to have spending their time on that when there were more important things they could do for us, like rolling out new services.

Chris: But what about the users while things were being changed?

Jean: They hadn't even noticed! My biggest headache is our accounting systems: they make the resource location visible at your level—but nobody else cares.

Chris: You didn't even have to come up with this solution yourself?

Jean: Nope. I didn't do a thing. ***eOS did it all!***[1]

[1]. We briefly toyed with calling our program *"e-kiN"* for the slogan *"it just did it!"* but we decided that the Nike Corporation might not be amused. So now we call it *eOS*, after the Greek personification of the dawn.

1 The resource economy

The proliferation of computers and the Internet into all aspects of commerce and society is well under way. Many of the fundamental technical issues to do with the components of modern computing systems are either solved, or well in hand. It is our position that the next wave of innovation—and hence research opportunities—lies in the field of aggregating pools of computing resources in support of the ex1ypplosion in scale, complexity, and diversity of computing services. The eOS program at HP Labs is targeting the technical barriers to this happening.

Besides the technical facilities implied by the scenario described above, what else will have to happen to make this vision become real?

- Resources must become fungible: which in turn implies that services (and their users) must be willing to trust that the resources they obtain from a shared pool cannot be tampered with by others—including their competitors.

This trust can be acquired in a number of ways. One approach is developing clever—but computationally expensive—cryptographic solutions for operating in a largely untrusted infrastructure (e.g., [Kubiatowicz2000] and [Farsite2001].) Instead, we believe that commercial systems will be able to benefit significantly from exploiting a much greater degree of trustworthiness of IDCs than the OceanStore project proposes. Today's "private intranets" is a classic example of this: the telcos are already providing the illusion of a dedicated network resource from their shared infrastructure.

- Data center computational resources must become "liquid commodities", and global markets develop for them.

This seems likely: electronic business exchanges for a wide variety of other goods are becoming of considerable importance, and computing resources are a natural application for this technology.

- Pervasive performance upgrades to the public internet and private intranets must be able to eliminate the need for most real local computation (but not storage, and not PCs).

This seems to be happening apace. Although the "last mile" looks likely to remain expensive for a while longer, long-haul networking is becoming quite cheap: indeed, the percentage of dark optical fibre is remaining roughly constant, thanks to the deployment of wave-division-multiplexing.

- Outsourcing and resource consolidation into large, concentrated *internet data centers* (IDCs) will continue.

This trend is favored by the economies of scale associated with internet access, security, power, cooling, and management that such centers represent.

- Resource demands will fluctuate faster than any single supplier can handle.

The Internet represents an effectively infinite potential load for almost any service, so a solution that can exploit a federation of resource providers will be necessary. In fact, we believe that it will be necessary for the eOS solution to manage resource demands that scale over roughly a 500 000:1 range—i.e., the range from one processor to about 10 data center's worth of demand for a single service. Ultimately, the only architectural limit should be the amount of silicon connected to the Internet.

- Unmodified applications must continue to be supported.

Although new applications and service types may spring up that can better exploit the new resource model, experience has taught us that backwards compatibility is the key to a successful adoption.

- Increasing the cost-effectiveness of resource provisioning by a factor of about 2 over the best-available solutions today will be sufficient to get this kind of solution adopted.

Although large, this represents a huge improvement on the few percent advantages that are often reported for individual technology improvements at the computing component level. Of course, the benefit will have to continue to scale up as the underlying technology becomes faster and more cost-effective: that is, it must be a multiplicative benefit on top of Moore's-law changes in computing and equivalent improvements in storage and networking, not a one-time addition.

- Finally, success will come to those businesses (and researchers!) who can provide stable, predictable business responses in the face of rapidly-fluctuating customer demands, at reasonable cost.

This has been supported by our research work in storage-system management [SSP2001], which began with a strong emphasis on cost minimization, and later discovered customer preference for predictable responses to requirements—even at the expense of greater system cost.

2 The eOS architecture

eOS is not a single artifact: it is better thought of as a set of related research activities that are striving towards a common vision over the next few years.

It should be emphasized, however, that it is possible to make progress towards an eOS system without requiring all of the aspects to be addressed. As working in the storage management domain has shown, solutions to portions of the lifecycle can be rolled out without the entire thing being in place.

Occasional examples from HPL's storage management work are used in what follows to illustrate that the eOS ideas already have antecedents in at least one field of endeavor [SSP2001].

2.1 Principles

To help achieve convergence, we've identified some underlying principles that we believe are fundamental to all our work, and devised an overall architecture that divides up the problem space in a way that will allow contributions from many sources to be integrated into the larger whole.

- IP will be the network service abstraction required by all services, including high-speed, local storage access.
- Heterogeneity of hardware, software and operating systems across resources is a given.
- Federated, nested structures are the only way to manage the required scale and business demands.
- *Predictability* is the single most important business metric to achieve—it is even more important than cost. This means

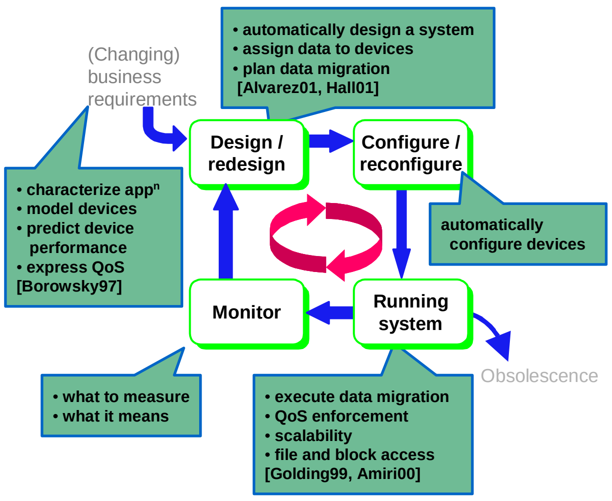
that QoS guarantees, or service level agreements (SLAs) are necessary.

The functions performed by the eOS architecture can be grouped roughly as follows:

- *Service-to-resource mapping*: what resources are needed by this service? when are they needed? and where?
- *Resource provisioning*: determining where to obtain and how to allocate the resources (planning), and how to set them up for use (configuration).
- *Monitoring, enforcement*: ensuring that security, QoS, etc. goals are met; monitoring the service level agreements; and deciding what should change as a result of this data.

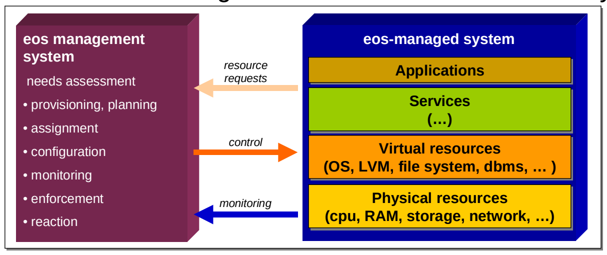
2.2 The eOS system life cycle

We have organized our architecture according to a lifecycle-based taxonomy. We first developed this concept in the storage-management space [Alvarez2000], where it has proven effective at providing insight into how to design tools that can support each stage effectively.



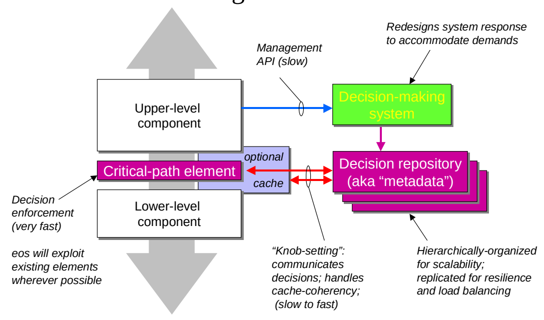
The figure above illustrates this lifecycle, and annotates it with some examples of the research work we have been performing in the storage-system-design space recently.

2.3 The eOS management architecture as a control system



By design, eOS works hard to avoid adding new APIs in the critical path: it manages existing services and resources, augmented by an eOS agent/control interface if necessary. This is essential if it is to be able to handle the diversity of heterogeneous systems that will continue to be the norm.

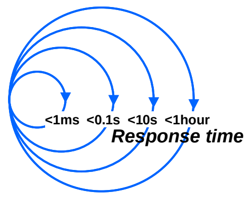
A management architecture needs to have points at which it can execute control—or else it cannot manage. To this end, our goal is to develop mechanisms that either exploit existing control points in the critical path, or insert new, minimalist ones. The diagram below represents the relationship between the control point and the “knob-settings” that tell it what to do.



Management API (slow)
Redesigns system response to accommodate demands
Upper-level component
Decision-making system
optional cache
Critical-path element
Decision repository (aka “metadata”)
Decision enforcement (very fast)
Lower-level component
eos will exploit existing elements wherever possible
“Knob-setting”: communicates decisions; handles cache-coherency; (slow to fast)
Hierarchically-organized for scalability; replicated for resilience and load balancing

The eOS architecture is designed to monitor the overall system behavior, and adjust the knob settings at appropriate intervals (this is where control theory comes in handy).

This architecture applies at many timescales: upper-level “policy decisions” are enforced by lower-level “mechanisms”—and become mechanisms for enforcing yet higher-level policies. This applies all the way out to the business-cycle scale.



In some cases, the knob settings are so large that they have to be cached (e.g., the metadata layout for a large storage system).

eos resource architecture

- **pools**
- allocation
- configuration
- service mgmt

2.4 The eOS resource architecture

The eOS architecture is divided it up into the four topic areas listed on the left.

2.4.1 Resource pools

A resource pool provides a scalable set of abstract resources, built from the physical resources of one or more data centers, and capable of being deployed against a wide range of performance, reliability, cost and security constraints. Resource types include:

- Computation (CPU, RAM, I/O cards, reconfigurable CPUs, …)
- Storage (on-line, off-line; local, remote; caches, …)
- Networking (internal, private intranet and public Internet; SAN; DNS, …).

2.4.2 Resource allocation

This is the process of deciding how many resources of what types, configured in which ways, will be needed to meet a set of service needs, and where those resources should be obtained from.

The process can be viewed as a kind of (NP-hard) bin packing, made more difficult by the desire to attain business goals, such as maximizing the cost-effectiveness of the resource deployment, balancing the spare capacity to handle short-term overloads better, and leaving maximum flexibility for future demands. Inputs to this process include:

- Service descriptions; performance requirements for the services at the service level (SLAs), and business goals.
- Constraints on the solution, such as geographic restrictions, and legal, security, and availability requirements.
- Existing resource allocations, and information on recent changes in requirements, environment, etc.

Actions that are taken in response to these inputs include:

- Mapping the service requirements into the resource needs they represent.
- Choosing possible allocation plans, taking the inputs into account.
- Negotiating with (potential) resource suppliers, including the local resource pools, and choosing which plan to apply.

2.4.3 Resource configuration

Given the resource allocation (or “design”) that comes out of the resource allocation step, the goal of resource configuration is to establish that design in a running system. The two steps are separate to allow them to operate on different timescales, and because resource configuration usually proves to be a somewhat simpler problem than resource allocation.

The input to the configuration step is a description of desired end state, together with access to the current configuration. (That is, resource configuration tools are used both for initial configuration and to changing an existing setup.) The first task of this step is to determine what changes are needed: “configuration diff” if you like. The actions taken then are a function of the type of resource being configured:

- For compute resources: load software (OS image, applications, …); and set the configuration (IP address, accounts, …).
- For network resources: configure switches, routers, firewalls, etc. to create secure network partitions
- For storage resources: configure disk arrays; SAN switches; host LVMs; file-system mount points; access paths, etc.)

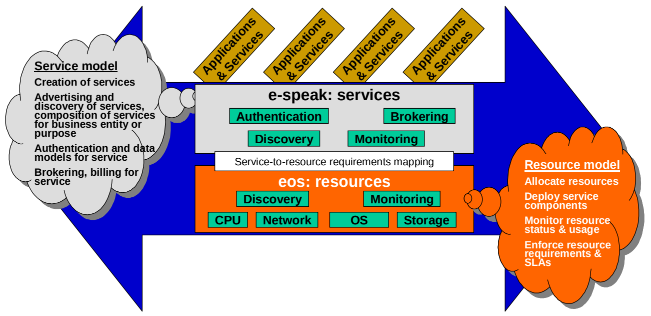
That some of these configuration problems are technically challenging in their own right is demonstrated by our work on the storage migration problem [Hall2001].

2.4.4 Service management

The steps described above operate primarily at the resource level. But since the intent of eOS is to support services, it has to have mechanisms to reason about them, and the effects that its decisions (are going to) have on those services. To achieve this, eOS maintains *models* of the services it supports. These service models are used for service-lifecycle management; establishing new service instances; provisioning for the services; and for monitoring the service behaviors (including service level objectives, failure detection and recovery).

The models describe how the service is composed from a collection of components; the resource demands of the service, and how these scale with service load; and data on interactions between service instances. Ideally, of course, these things will be known *a priori*. But life is rarely like that, and the very diversity of service types will make it unlikely to be achievable in any case. As a result, a significant research topic in eOS is determining such service properties experimentally, while they are “live”.

The HP e-speak architecture [Karp2000] is one example of a system that manages services; eOS manages the resources that those services execute on. We are betting that the provision of a managed resource pool at the eOS level will be at least as pervasive as scaling services by means of service-level aggregation. We expect that the resource-level approach will


Service model
Creation of services
Advertising and discovery of services, composition of services for business entity or purpose
Authentication and data models for service
Brokering, billing for service
Applications & Services
Applications & Services
Applications & Services
Applications & Services
e-speak: services
Authentication
Brokering
Discovery
Monitoring
Service-to-resource requirements mapping
eos: resources
Discovery
Monitoring
CPU
Network
OS
Storage
Resource model
Allocate resources
Deploy service components
Monitor resource status & usage
Enforce resource requirements & SLAs

scale to a wider range of service types with less implementation effort, although specialized services (such as email) may be able to provide their own management as well.

2.4.5 Virtual resource management

Many eOS systems will be deployed in the presence of existing management entities. A high-level example from the storage domain is [StorageNetworks2001], but low-level examples also exist, such as the HP AutoRAID product, which manages its own storage hierarchy internally [Wilkes1996].

A useful way to think of these entities is as providers of new, nested resource types, with their own—typically rather richer—behaviors than stand-alone physical resources. eOS simply treats them accordingly.

Similarly, an important—and so far neglected—research thrust for eOS is in the area of federated services. eOS systems will have to negotiate with their peers to obtain resources to meet their own obligations. This is complicated by trust, administrative, and financial boundaries—but a solution will represent a big step towards the resource economy [Bhoj1999].

eos research questions

- **pools**
- allocation
- configuration
- service mgmt

3 Key research questions

The eOS problem area represents a huge opportunity for high-quality, interesting research work. And we are not alone in believing this: several other research teams are looking in similar directions (e.g., UCB OceanStore [Kubiatowicz2000] and Ninja [Gribble2001], Microsoft [Farsite2001], and IBM [Océano2001]).

In the space that remains, the rest of this position paper provides the briefest of outlines of just some of the highest-impact research topics in the eOS domain. The list is not meant to be exhaustive, but to give a flavor of the types of issues that it suggests be addressed.

3.1 Resource-pool management

Before resources or services can be managed, they must be found and their properties determined (e.g., security, performance, reliability, …)—a process known as *discovery.* In use, their behavior must be monitored (e.g., for performance, utilization, availability, …). All this has to be done automatically, in the face of resource and service types that have not yet been met, partial system failures, and occasionally disconnected operation.

To provide control, appropriate control points (*actuators*) need to be located or inserted; and appropriate parts of control-theory need to be developed for the kind of hands-off, nested cycles that resource management requires.

The need for business predictability translates directly into a need to enforce QoS goals and SLAs, as well as requiring more work in the economics of QoS-based system management (e.g., in appropriate pricing mechanisms for controlling user behavior in an environment where there is effectively infinite demand).

We will need new end-to-end security systems for secure virtual environments, VLANs, OS partitions, source encryption, key distribution, payments, … and so on.

3.2 Resource allocation

Before a service can be begun, the set of resources it needs must be known, more or less. This must be a prediction, derived from mappings from service-level demands to resource demands. Such mappings need building, validating, and tuning from prior service executions—automatically. Self-tuning, robust statistical/econometric models will also be necessary for predicting future service demand trends.

Determining the "best" way to allocate resources to meet service demands requires the ability to search a space of possible allocations, and predict their effects, taking into account business objectives and constraints (time, scale, security, availability, QoS, …). We hope to leverage our work in this area for block-level storage systems (e.g., [Borowsky1997]) to the wider eOS problem domain.

The process of obtaining resources will require automated, multi-site provisioning negotiations, across a federated set of resource providers.

3.3 Resource configuration

Although setting up a set of resources to meet a designed configuration may be simpler than designing the configuration, it still not a trivial problem—especially at the scales we are contemplating, and in the face of requirements that the system remain operational during changes that must happen in a coordinated manner across multiple sites, with little or no human involvement, across a heterogeneous range of resource and service types, and in the face of near-arbitrary failures [Golding1999]. These are all hard problems. Their confluence is particularly interesting.

3.4 Pervasive architectural issues

We think the best way to manage services (and resources, for that matter) is to start with a rich, extensible, common description model that encompasses everything that is used across the entire eOS suite: demands, services, resources, configuration constraints, discovery, current state, changes, deployment, life-cycle, SLA etc. Given such a model, it is possible to build tools that each perform one function: that is, the model is the moral equivalent of the text streams that made UNIX pipes so effective a composition mechanism.

In the storage-management domain, our own modeling system (called Rome) handles some of these issues: it can represent *goals* (for quantifying the QoS/SLA requirements, objectives, and constraints); *system descriptions* (for capturing the current state, and capabilities); *models* (for predicting the future); *measurements* (to capture system behavior); and *changes* (plans and techniques for affecting the result).

4 Conclusions

The eOS vision builds on existing research work in a wide variety of areas. We think it represents a compelling, practical vision of what a future resource economy could be like, offers an architecture that we think will get us there, a path that allows incremental deployment in real systems, and is a rich source of research topics whose solutions will be of vital practical importance.

We hope you agree—and we are actively seeking research partners to help make the vision reality. Just remember the slogan: ***eOS did it all!***

Acknowledgements

Many people contributed to the ideas that form the basis of this paper including: Rich Friedrich, John Manley, Patrick Scaglia, John Sontag, Hans Stork, Simon Towers, Alistair Veitch, and the past and present staff of HPL's Storage System Program.

References

[Alvarez2000] Guillermo Alvarez, Kim Keeton, Arif Merchant, Erik Riedel, John Wilkes. *Storage systems management*. Invited tutorial presented at SIGMETRICS 2000 (Santa Clara, CA, June 2000). Available from: http://www.hpl.hp.com/SSP/papers .

[Bhoj1999] Preeti Bhoj, Sharad Singhal, and Sailesh Chutani. *SLA management in federated environments*. Proc. 6th IEEE/IFIP Intl. Symp. on Integrated Network Management, (Boston, MA, May 24-28, 1999). Available at http://www.hpl.hp.com/techreports/98/HPL-98-203.html .

[Borowsky1997] E. Borowsky, et al. *Using attribute-managed storage to achieve QoS*. Presented at *5th Intl. Workshop on Quality of Service*, (Columbia Univ., New York, June 1997). Available at http://www.hpl.hp.com/SSP/papers

[Farsite2001] Microsoft Research. *Farsite: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment.* http://www.research.microsoft.com/sn/Farsite/ (Jan. 2001).

[Golding1999] Richard Golding and Elizabeth Borowsky. *Fault-tolerant replication management in large-scale distributed storage systems.* Proc. 18th IEEE Sumposium on Reliable Distributed Systems (SRDS'99) (Lausanne, Switzerland, 19-22 October 1999).

[Gribble2001] Steven D. Gribble, et al. *The Ninja architecture for robust Internet-scale systems and services.* To appear in *Computer Networks on Pervasive Computing*; available from http://ninja.cs.berkeley.edu/pubs/pubs.html (Jan. 2001).

[Hall2001] Joseph Hall, Jason Hartline, Anna R. Karlin, Jared Saia and John Wilkes. *On algorithms for Efficient data migration.* Proc 12th ACM-SIAM Symposium on Discrete Algorithms (SODA), Jan. 2001 (Washington, DC).

[Karp2000] Alan Karp. *E-speak E-xplained.* HP Labs technical report HPL–2000–101, Aug 2000. Available at: http://www.hpl.hp.com/techreports/2000/HPL-2000-101.html

[Kubiatowicz2000] John Kubiatowicz, et al. *OceanStore: an architecture for global-scale persistent storage.* Proc. 9th ASPLOS, pp. 190–201 (November 2000).

[Océano2001] IBM Research. *The Océano project.* http://www.research.ibm.com/oceanoproject/ (Jan. 2001).

[SSP2001] HPL Storage Systems Program. http://www.hpl.hp.com/SSP/ (Jan. 2001).

[StorageNetworks2001] Storage Networks, Inc. http://www.storagenetworks.com/content/home/ (Jan. 2001)

[Wilkes1996] John Wilkes, Richard Golding, Carl Staelin, and Tim Sullivan. *The HP AutoRAID hierarchical storage system.* ACM Transactions on Computer Systems **14**(1):108-136, February 1996.