Y ¾ Pervasive Services Infrastructure

Dejan Milojicic, Alan Messer, Philippe Bernadat,
Ira Greenberg, Guangrui Fu, Olaf Spinczyk[1],
Danilo Beuche[1], Wolfgang Schroder-Preikschat[1]
Computer Systems and Technology Laboratory
HP Laboratories Palo Alto
HPL-2001-87
April 4th , 2001*

E-mail: [dejan, messer, bernadat, iragreen, guangrui]@hpl.hp.com,
[olaf, danilo, wosch]@ivs.cs.uni-magdeburg.de

pervasive, services, infrastructure, Java, wireless

Future systems have been characterized as ubiquitous, pervasive, and invisible. They will consist of devices that are diverse in size, performance, and power consumption. Some of these devices will be mobile, posing additional requirements to systems software and applications. The focus will move from technology to deployment and ease of use of services. Consequently, traditional paradigms for reasoning about, designing, and implementing software systems and services will no longer be sufficient.

We believe that this future vision will rely on a three-tier infrastructure consisting of back-end servers, infrastructure servers, and front-end clients (mobile or static, handheld or embedded). The critical question for future systems will be how to deliver services on-demand from back-end servers to resource-constrained clients. If we can handle the new requirements of these systems, we can enable this computing infrastructure to offer significantly more services to users in a more pervasive way.

* Internal Accession Date Only Approved for External Publication
[1] Otto-von-Guericke-Universitat Magdeburg,Universitätsplatz 2, 39106 Magdeburg, Germany

Ψ—Pervasive Services Infrastructure

Dejan Milojicic, Alan Messer, Philippe Bernadat,
Ira Greenberg, and Guangrui Fu
Hewlett-Packard Labs
[dejan, messer, bernadat, iragreen, guangrui]@hpl.hp.com

Olaf Spinczyk, Danilo Beuche, and
Wolfgang Schröder-Preikschat
Otto-von-Guericke-Universität Magdeburg
[olaf, danilo, wosch]@ivs.cs.uni-magdeburg.de

Abstract

Future systems have been characterized as ubiquitous, pervasive, and invisible. They will consist of devices that are diverse in size, performance, and power consumption. Some of these devices will be mobile, posing additional requirements to system software and applications. The focus will move from technology to deployment and ease of use of services. Consequently, traditional paradigms for reasoning about, designing, and implementing software systems and services will no longer be sufficient.

We believe that this future vision will rely on a three-tier infrastructure consisting of back-end servers, infrastructure servers, and front-end clients (mobile or static, handheld or embedded). The critical question for future systems will be how to deliver services on-demand from back-end servers to resource-constrained clients. If we can handle the new requirements of these systems, we can enable this computing infrastructure to offer significantly more services to users in a more pervasive way.

1 Introduction

The future of computing has been painted by many visionaries. It was coined as ubiquitous computing by Mark Weiser [31], and D.A. Norman introduced invisible computing [22]. IBM promotes pervasive computing [11], Sybase calls it mobile embedded computing [28], and Sun uses the term Post-PC era [27]. HP Labs' vision is presented in CoolTown [14]. Several umbrella projects in major universities are also exploring these topics, such as Aura at CMU [21], Portolano at the University of Washington [7], Endeavour at Berkeley [10], and Oxygen at MIT [6]. The government is investigating Ubiquitous Computing [4] and Composable High Assurance Trusted Systems [5]. Finally, there are numerous startups in this area, such as StreamTheory [26], Transvirtual [29], and WordWalla [32].

Common to most of these visions are the ideas of blending computers into the infrastructure and providing user-friendly services to non-expert users. The center of gravity is moving from technology to users and services. Mobile and wireless are becoming common rather than the exception. Connectivity and bandwidth are improving, approaching 5-20Mbps for 4G networks in 2005. We believe that the focus of future technology will be in the intersection of the Internet, on-line services, and mobile wireless communication. The environment will consist of globally distributed high-end servers hosting services (e.g., Oceanstore [15]), mid-point servers caching and otherwise complementing service delivery to clients (e.g., Akamai), and a variety of client devices.

Under services, we assume a variety of applications and underlying support (description, look-up, storing state, etc.). Examples include traditional desktop applications, enterprise applications (e.g., project management, expense reporting), personal information management, and various vertical market applications, such as retail, health care, financial, entertainment, and travel.

We are investigating a Pervasive Services Infrastructure (PSI—Ψ in Greek) for delivering Internet services to (wireless) users. The Ψ vision is "Any service to any client (anytime, anywhere)". We envision that in the future it will be possible to deliver services to clients on their mobile, handheld devices in the same way as it is possible at the desktop today. In addition to the traditional challenges of mobile computing [25], we believe that the biggest challenges of this environment are in adapting services to diverse client devices and in delivering services to clients.

Most devices are resource-constrained compared to desktop systems. They differ in many ways, such as user interfaces, CPU, memory size, and power constraints, all of which will require some form of adaptation of the service delivery. We are investigating how offloading parts of applications to *mid-point servers* can enable and enhance service execution on a resource-constrained device. Dynamic delivery of services to devices is required to eliminate the need for pre-installed services, to enable the downloading of dynamically composed services, and to support system evolution. For Ψ, client devices and infrastructure act as caches for delivering services from back-end servers, thereby improving the performance of remote access.

The rest of the document is organized as follows. Section 2 discusses adaptive offloaded services. Services-on-Demand is described in Section 3. In Section 4, we present some initial results. In Section 5, we compare our project to related work. We summarize the paper and present future work in Section 6.

2 Adaptive Offloaded Services

Pervasive systems bring a proliferation of devices and infrastructure with differing capabilities and capacities.

We believe the scale and diversity will lead to several problems in supporting services on these devices. Consider Personal Digital Assistants (PDAs). While fundamentally doing the same task, specifications have varied greatly: color screens, faster processors, memory capacities. Providing even a simple service to these similar devices presents a complex task for the software manufacturer who must provide software for the lowest common denominator or must provide separate versions. When computing is pervasive or multiple services are run simultaneously, this problem becomes more acute.

We believe such device limitations can be relieved when devices are universally networked. Instead of hitting resource constraints when supporting a service, other parts of the infrastructure could cooperate to offload parts of the service from devices. However, offloading services in a networked environment is difficult, because the available device resources and each device's location may change dynamically. Therefore, we believe that if services are to best exploit this environment, they will also need to adapt to changes in it.

Scalability also poses interesting problems when providing services to so many devices. With just three to five networked devices per user and several million users, the problem of running complex services such as multimedia or interactive services can easily outgrow a single central server cluster. Proxy caches and companies such as Akamai have shown that a multi-tier approach can be used for data to overcome this problem of scale. We believe that service provision can similarly benefit by a multi-tier approach for caching and execution, allowing service offloading into the infrastructure.

To illustrate our vision, consider using a PDA to edit a digital photograph sent to you by URL. This poses a problem for your device, because despite having a color screen there is little spare capacity. Yet, you would like to edit the photograph even if you cannot fully execute an editing application, such as Adobe Photoshop. Instead, the device's runtime uses its understanding of the service, its surrounding devices, and back-end servers to allow the device access to the service, independent of the limitations of the device. For example, the main execution may be performed on a nearby server leaving the device to handle performance sensitive operations and I/O (see Figure 1). A more capable device might run the main execution and user interface, and use the server to hold swapped memory and to perform intensive image processing.

We believe that dividing service responsibility can be achieved by borrowing resources from mid-point servers (e.g., memory swapping), or by constructing the service from multiple components that are placed and executed separately. Services developed in compositional frameworks are therefore good candidates for our vision. Unfortunately, except for very coarse granularity, few services have been constructed this way. However, we believe that some automatic decomposition may be possible using additional system support in modular systems (e.g., Java).

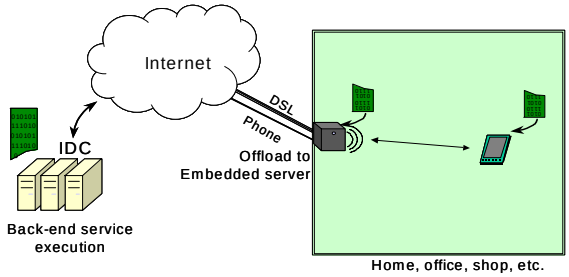

Figure 1 Adaptive Offloaded Services

We believe that infrastructure support for service offloading and adaptivity will allow scalable, high-performance services for a multitude of differing devices in a mobile environment. However, several questions will have to be answered. What service framework requirements would be needed for such distributed execution? Can existing services be automatically split and efficiently placed? Can placement be performed transparently or should services and runtimes interact? Can services be performant and scalable when distributed across a multi-tier environment? Can service offloading effectively cope with the widely varying characteristics of different devices? Can users roam and still effectively obtain service? What if the service infrastructure decides to migrate parts of the service onto another node, or there are communication errors? If operation is not transparent, it may be necessary to do a service-specific cleanup, such as closing temporary files or restarting, requiring extra application code.

Initially, we will investigate the division, placement, and execution and of support for services. We will manually divide a service's storage and execution across a three-tier environment (server, mid-point and client) to study the performance and scalability effects of distributed execution with an educated split. At the same time, we will examine the characteristics of services (active memory footprint/access patterns, execution paths) to determine whether manual or automatic splitting is appropriate for services. Using these results, we will investigate the effect of roaming in such an environment and the effects of placing the same service on alternative resource-constrained devices.

3 Services-on-Demand

It is becoming increasingly important to provide services-on-demand with minimal support from users. In

general, users will be less computer savvy and will want to focus on using services, not administering them. Users will connect to the Internet with a wide variety of devices running a diverse collection of system software (operating systems). Many services will become available and will be updated frequently. Users will want to access their preferred services and environments from any location and any device.

These trends call for a new system software infrastructure to support services-on-demand. In this approach, no "a priori" service installation will be required. Instead, desired services will be dynamically located and retrieved, perhaps with service brokers, based on available resources. A trust framework will be needed so that services with an appropriate level of trust can be obtained, and services with different levels of trust can work together. Appropriate billing models will have to be integrated into the infrastructure so that clients can be dynamically billed for the services they use. Because user devices will have volatile storage, user environments and data will have to be securely and persistently stored on storage servers, with support for privacy and integrity. Services, user environments, and data will have to be enabled and loaded on a user's device wherever the user is located, even if the user is moving. The infrastructure should also support disconnected operation for services that can operate in that manner.

Consider the following scenario. While listening to the news, you hear about a new financial service that makes it possible to simulate some investment model. You connect to the internet, request the service, and start it. The service might also be located by type from a look-up service provided by a service broker. As opposed to a Web-server-based service running remotely, the service may run partially or entirely on your device. This is similar to Java applets. In reality, the service can be any type of Java software, and it will be seamlessly installed (or cached) on your device, if it fits. Otherwise, it will be offloaded to a support server.

Assume that the service saves simulation results as files. Your data is managed by a storage provider, and the files are transparently updated there, at reconnection time if required. You decide to move and will either carry your PDA or use another device at your destination (hotel, airport, etc.). When you log in on this new device, you will provide a user key to the storage provider. The service will be downloaded (if it is not already cached), and your private files and environment will be retrieved (see Figure 2).

Achieving these goals poses several fundamental challenges. How should device resources be characterized (primary and secondary storage, user interface, peripherals, etc.), and how should service resource requirements be characterized? How can services be acquired and composed with appropriate levels of trust? What service look-up strategies make sense, and how should users be billed for services? What service-dependent reconciliation strategies can be used for disconnected operation?

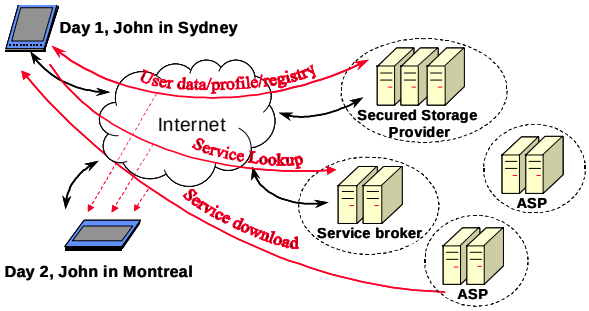

Figure 2 Services-on-Demand Infrastructure

Our plan is to first build a simple prototype for the Java Virtual Machine to investigate performance. Switching to system software that noticeably degrades performance is unacceptable. We want to determine the real cost of using a virtual machine and downloading services-on-demand, and to see how much caching helps. We will experiment with the infrastructure devices that are small, portable, resource-constrained, and JVM-enabled. In the long term, we will investigate resource characterization and disconnected operation, and provide a security and trust framework.

4 Preliminary Performance Measurements

To investigate service offloading, we explored the benefit of offloading parts of the Java runtime libraries. We used jPure [3] on top of the Pure OS [1]. We compared the EmbeddedCaffeineMark benchmark on a variety of JVMs and Java-to-native compiler-based solutions with a static offloading of the runtime libraries using jPure. Table 1 illustrates that a fixed offloading of runtime libraries can offer a footprint/performance trade-off. Because jPure is a distributed system, its memory is distributed on a client and server.

Table 1 Java Execution Environment Comparison

Java environment vs consumption & score	Total initial memory consumption (KB)	CaffeineMark overall score	kB per score point
Sun JDK 1.3 (JIT)	7476	1828	4.1
IBM JDK 1.3 (JIT)	8212	5155	1.6
GCJ-Linux 2.95	1416	3109	0.5
jPure (client+server)	312+800	2167	0.5

Next, to determine the dynamic behavior of such Java executions, we modified Kaffe to record the thread and heap pages (4kB) touched for each 10,000 bytecodes executed. Figure 3 indicates that for a 1538-page heap

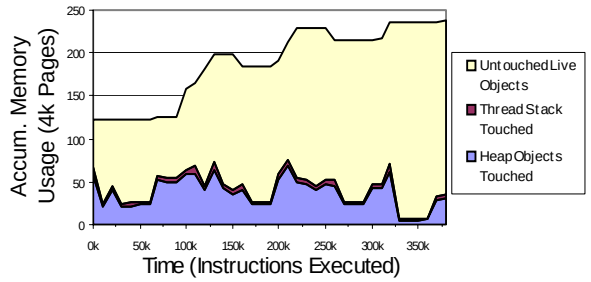

Figure 3 The Java heap page working set of 'Hello World'

with between 125 and 225 pages of live objects only between 6 and 70 pages are actually accessed. This simple example indicates that there are possibilities for dynamic offloading of memory for services, because for very few pages are actively used at the same time.

To investigate the cost of downloading services on demand, a prototype system was instrumented to measure the elapsed time until a service is ready. Table 2 shows the performance of various service classes, ordered by download size. All services are downloaded from servers outside of the HP intranet. The reference is measured when the service is fully cached. The download overhead is only incurred when the service is first used. The relative performance (in addition to reference time) is measured uncached for three connection types:

- **Intranet** within HP (with Web proxy side effects).
- **Wavelan** connected to the HP intranet.
- **Dialup** telephone line.

Table 2 Services-on-Demand (SoD) Loading Overhead

Service	Local Loading (s)	SoD Loading (s)			Loaded URLs		Java Class	
		Intranet	Wave	Dialup	Cnt	Size (B)	Cnt	Size (B)
calculator	0.1	~0	+0.1	+4.7	2	9520	2	9520
calendar	0.2	~0	+0.4	+4.7	4	12952	4	12952
editor	0.2	~0	+0.3	+5.2	7	15885	7	15885
game	0.2	~0	+0.3	+6.1	9	18739	9	18739
agenda	0.2	+0.1	+0.7	+8.4	1	35360	12	59846
ftp	0.6	+2.1	+3.2	+20.9	3	107725	22	142155
mail	2.5	+2.7	+15.1	+129.4	1	675046	138	365574

These preliminary results indicate that, on a reasonable network link of the future, the overhead of downloading and installing services on demand is relatively small (a couple of seconds). Even for large applications that can be compressed in the JAR format, performance is not unreasonable. However, we can see that for this application (mail) the code size is only 300KB leaving the rest as documentation and graphics. A service-on-demand version of this application might download these supplemental items incrementally or they may be cached to avoid noticeable delays. Overall, these results point towards the suitability of Java applications coupled with a good infrastructure to effectively present services on demand without adversely affecting performance.

5 Related Work

Due to limited space, we do not discuss related work in detail; instead we just compare it with our project. In addition, we omit some of related work, such as Rover [12], Coda [19], and GAIA [24].

The **Odyssey** project defines a software platform for application-aware adaptation of diverse mobile applications [21]. This approach considers agile applications in varying fidelities, adapting to system variations, e.g., in network bandwidth. Our work takes a similar view to adaptability applied to services and the system. We are interested in adaptability in a different scope, such as adaptability using computation and storage placement.

The **Ninja** project (as well as earlier work by Fox [8]) investigates a software infrastructure for next generation Internet services [10]. Services are designed to be composable, customizable, and accessible from a variety of device types. The service components can be executed closer to the client to enable transcoding. Our approach takes locality of computation further by considering mid-point servers as locations for computation and storage used by service providers and service clients. In addition, our focus is on offloading services rather than altering service fidelity as with proxy transcoding.

The **Oxygen** project studies software environments for composable applications and systems [6]. Their approach is to use abstraction, specification, persistent storage, and transactions to support change through adaptation and customization. The **Portolano** projects (Active Fabric and ARCaDE) focus on service provisioning for self-organizing, mobile, composable services; service migration; and automatic service management [7]. Oxygen and Portolano take an active networks approach. We instead consider the computation and storage in the infrastructure to be temporary service caches, with services ultimately originating from back-end service providers.

To support nomadic users, HPL's **CoolTown** project offers a model based on a convergence of Web technology, wireless networks, and portable devices [14]. CoolTown attempts to bridge physical and virtual worlds, whereas Ψ addresses resources and services. CoolTown addresses location dependency and connectivity, while Ψ emphasizes deployment and disconnection.

There are also related industrial standards. Universal Description, Discovery, and Integration (**UDDI**) is a specification that defines a way to publish and discover information about services [30]. Open Services Gateway Initiative (**OSGi**) explores Java platform independence and dynamic code-loading for small-memory devices [23]. Ψ can benefit from either standard.

6 Summary and Future Work

We presented our vision of a pervasive services infrastructure. In particular, we addressed the two technologies required to achieve our vision: adapting services to execute on resource-constrained devices and installing services on demand. We believe that both are required to achieve ubiquitous systems. We also presented preliminary results indicating the benefits of offloading and downloading services. If the Ψ vision can be achieved, it may be possible to offer today's desktop services to a variety of resource-constrained devices in a mobile environment.

We are also interested in investigating service composition. Promising techniques in this area are component-based computing [20] and aspect-oriented programming [13]. The ideal service would consist of a number of components glued together, and would separate placement and configuration from the core functionality, such as in Regis [18]. The service component boundaries would provide points for switching between local and remote execution control of the infrastructure runtime. Java RMI/JavaBeans, Puppeteer [16], and CANS [9] are other examples of component-based systems. Aspect-oriented programming takes a similar approach [13]. A tool called the aspect weaver can be used to connect the components with the implementations of their technical aspects. The result is a loose coupling, which leads to a high degree of configurability at either compile- or runtime. Examples include D [17] and work by Becker [2].

Acknowledgments

We are indebted to G. Candea, C. Karamanolis, K. Keeton, E. Kiciman, M. Mahalingam, D. Muntz, G. Snider, and J. Wilkes for reviewing the paper and/or otherwise contributing to the project. Their comments significantly improved the content and presentation.

References

[1] Beuche, D., et al., "The PURE Family of Object-Oriented Operating Systems for Deeply Embedded Systems," Proc. 2nd IEEE Symp on OO Real-Time Dist Comp, StMalo, France, May 1999.

[2] Becker, C., Geihs, K., "Quality of Service — Aspects of Distributed Programs," ICSE'98 Workshop on Aspect-Oriented Programming, 1998.

[3] Beuche, D., et al., "JPure - Purified Java Execution Environment for Controller Networks," Proc. of the IFIP Workshop on Dist. and Parallel Embedded Systems, Paderborn, Germany, Oct 2000,

[4] DARPA ITO Ubiquitous Computing Program, www.arpa.gov/ito/research/uc.

[5] Composable High Assurance Trusted Systems (CHATS), www.arpa.gov/ito/research/chats.

[6] Dertouzos, M.L., "The future of computing, Scientific American," July 1999. http://oxygen.lcs.mit.edu.

[7] Esler, M., et al., "Next century challenges: data-centric networking for invisible computing: the Portolano project at the University of Washington," Proc of 5th ACM/IEEE Conf. on Mobile Computing and Networking, Aug 15-19, 1999, Seattle, WA. http://portolano.cs.washington.edu.

[8] Fox, A., et al., Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives," IEEE Personal Communications, August 1998.

[9] Fu, X., et al., "CANS: Composable, Adaptive Network Services Infrastructure", to appear at proc. of USENIX USITS, 2001.

[10] Gribble, S., "The Ninja Architecture for Robust Internet-Scale Systems and Services," Special Issue of Computer Networks on Pervasive Computing, 2000. http://endeavour.cs.berkeley.edu/

[11] IBM Pervasive Computing http://www-3.ibm.com/pvc/.

[12] Joseph, A., et al., "Building Mobile Applications with the Rover Toolkit," Proc. 15th SOSP, Copper Mountain Resort, CO, Dec. 1995, pp 165-171.

[13] Kiczales, G., et al., "Aspect-Oriented Programming," Proceedings of the ECOOP 1997, Finland. Also available as Xerox PARC, TR SPL97-008 P9710042, Feb., 1997.

[14] Kindberg, T., et al., "People, Places, Things: Web Presence for the Real World. ", Proceedings of the third WMCSA, 2000. see also HPL CoolTown, http://cooltown.hp.com/.

[15] Kubiatowicz, J., et al., "OceanStore: An Architecture for Global-Scale Persistent Storage", Proc. of 9 ASPLOS, Nov. 2000.

[16] de Lara, E., et al., "Puppeteer: Component-based Adaptation for Mobile Computing," to appear at proc. of USENIX USITS, 2001.

[17] Lopes, C.V., Kiczales, G., "D: A Language Framework for Distributed Computing", Xerox PARC, TR SPL97-010 P9710047, Feb. 1997.

[18] Magee, J., et al. "A Constructive Development Environment for Parallel and Distributed Programs," In IEE/IOP/BCS Distributed Systems Engineering, 1(5): 304-312, Sept 1994.

[19] Mummert, L.B., et al., "Exploiting Weak Connectivity for Mobile File Access," Proc. of the 15th ACM SOSP, Dec. 1995, Copper Mountain Resort, CO, pp 143-155.

[20] Nierstrasz, O., Gibbs, S., and Tsichritzis, D., "Component-Oriented Software Development," CACM, v 35, no 9, September 1992, pp. 160-165.

[21] Noble, B.D., et al, "Agile Application-Aware Adaptation for Mobility," Proc of 16 SOSP, St. Malo, France, October 1997. Aura projects at CMU http://www.cs.cmu.edu/~aura.

[22] Norman, D. A., "The invisible computer," Cambridge, MA, MIT Press, 1998.

[23] OSGI Service Gateway Specification, available at www.osgi.org.

[24] Roman, M., and Campbell, R.H., "Gaia: Enabling Active Spaces," Proceedings of the 9th ACM SIGOPS European Workshop, Kolding, Denmark, September 2000.

[25] Satyanarayanan, M., "Fundamental Challenges in Mobile Computing", Proc. of 15 ACM Symp. on Principles of Dist. Computing, May 1996, Philadelphia, PA, pp 61.

[26] StreamTheory, www.streamtheory.com.

[27] Sun Microsystem, "The .com Revolution Meets Consumer Appliances", available at: www.sun.com/990106/ces/.

[28] Sybase white paper, "Enabling e-Business Anywhere, Anytime: the Sybase Strategy," http://my.sybase.com/detail?id=1003164

[29] Transvirtual www.transvirtual.com.

[30] UDDI Technical White Paper, available at www.uddi.org.

[31] Weiser, M., "Some Computer Science Problems in Ubiquitous Computing," Communications of the ACM, July 1993, 75-84.

[32] WordWalla www.wordwalla.com.