Replication and Caching Strategies in Cadmium

Aline Baggio

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Replication and Caching Strategies in Cadmium

Aline Baggio

No 3409

Avril 1998

THÈME 1

Rapport de recherche

ISSN 0249-6399

Replication and Caching Strategies in Cadmium

Aline Baggio*

Thème 1 — Réseaux et systèmes
Projet SOR — http://www-sor.inria.fr/

Rapport de recherche n ° 3409 — Avril 1998 — 7 pages

Abstract: The Cadmium project aims at providing system support for disconnected and mobile users. This includes basic mechanisms for ensuring data availability, such as replication, caching or prefetching, as well as additional ones for consistency, adaptation to environment changes, etc. In order to insure good tradeoffs when degrading a service while disconnected, we provide resource-awareness and hooks for application or user-level interaction, along with automatic system support.

Key-words: Mobile computing, replication, cache, prefetch, Cadmium

(Résumé : tsvp)

* E-mail: Aline.Baggio@inria.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Stratégies de réplication et de mise en cache dans Cadmium

Résumé : Le projet Cadmium a pour objectif de fournir un support système pour les utilisateurs mobiles ou déconnectés. Il fournit des mécanismes assurant la disponibilité des données (tels la réplication, la mise en cache ou le pré-chargement), la gestion de la cohérence, l'adaptation aux changements de l'environnement, etc. Afin de garantir une dégradation de service acceptable pendant les déconnexions, Cadmium tient compte des ressources disponibles et autorise une étroite collaboration entre l'utilisateur, l'application et le système.

Mots-clé : Informatique mobile, réplication, cache, pré-chargement, Cadmium

1 Unplug-and-play

Laptop computers, cellular telephony for voice and data, indoor/outdoor wireless networks are now becoming a common alternative to classical fixed computing.

In such a context, the challenge of mobile computing is to provide users with mobile devices as useful as fixed ones. The notion of usefulness strongly depends on the main usage of the device, i.e. phone, calendar or notepad; as well as its size, CPU, memory, or storage capabilities. In many case, mobility implies reliance on the fixed part of the network for data storage, remote execution, or at least communication or access to fixed resources such as phones, printers and so on.

In the following of this article, we mainly consider the use of powerful laptop computers, as opposed to hand-held notebooks, keyboardless devices or smart cards. According to our views, laptop usefulness intends providing a stable and quite normal working environment despite changes such as relocation or network bandwidth varying from high-speed to none. Achieving this goal encompasses support for availability of data across disconnections as well as responsiveness to environment changes [1].

Disconnected work periods and laptop reconnection still raise far too many problems. One still can not find a global and integrated support to definitively get rid of manual data manipulations such as copy or management of concurrent modifications.

On one hand, applications have to integrate more support for mobile users. Browsers already enable minimal personal caching for off-line browsing; calendars provide simple resolution tools to manage concurrent updates. Nevertheless, this support is still very immature. For example, mail handling tools do not handle multiple mail folders, multiple source system mail boxes, nor concurrent access from multiple hosts.

On the other hand, many systems now include IPv6 support [4]. But they still lack the full range of modules and tools to take in account mobility at a very low level, including basic functionalities such as caching, conflict handling or environment monitoring [3, 6, 9].

In a few words, we consider that the "unplug-and-play" paradigm is still to come. This is not only dependent of the software but also need an efficient and ubiquitous wireless network infrastructure for data communications.

2 Rationale

Cadmium addresses the above problems by allowing data and code replication on both parts of the network, fixed and mobile; by providing additional support for dealing with these multiple copies; and by setting up hooks for system or application adaptation to environmental changes.

At a low level, Cadmium uses an object model. Objects have various granularities, from single piece of data to whole file or directory contents. They can be clustered in groups. Groups of objects provide an adaptable-grain scheme, and can be used as a replication unit. In addition, objects can be active, in the sense of holding code and threads for stand-alone execution. Finally, Cadmium makes use of an underlying support system that enables object migration from one process to another. Migration enable hosts to do remote executions on peers or servers.

At a higher level, we base our support on a cooperation between the system and the applications enabling resource-awareness [2]. This approach is inspired by the Odyssey adaptability [11, 9].

Resource-awareness allows applications to track evolution of availability, load, or state of other critical resources. Each application can select its own important resources, critical thresholds, adaptation criteria and get feedback from the system. Given this mechanism, an application can select the best way to degrade [9] or improve its service whenever required. It can also take benefit of each resource as the opportunity arises; i.e. networks, nearby hosts or equipments, equivalent local or nearby replicas. Successful experiments of this approach have been done with bandwidth use in weakly connected environments [8].

3 Strategies for replicating data

Based on its object model, Cadmium enables object replication; singly or in groups. By replication, we mean all algorithms based on data copying, such as caching, prefetching, etc. In most cases, the replicated objects are fully accessible to the applications, allowing writes as well as reads. The replicas can be stored directly in the application space or in a dedicated process space (i.e. a cache).

No cache: This strategy corresponds to a laptop without specific mobility support. If some data are needed while disconnected, the user has to handle the copy manually. This is painful and induces many problems, such as references towards the data, or consistency management between concurrent versions.

Systematic caching: This can be stated as the reverse approach. Each time an application tries to access a remote object, the system automatically creates a replica on the laptop and uses exclusively that replica. This has the advantage of supporting unpredictable disconnections since the cache already holds currently in-use objects. However, this does not guarantee that the user will not observe misses, and implies consistency management even while connected. Bayou [3] uses this kind of approach. The caching is achieved only once since Bayou replicates a whole database, and persists until an explicit deletion of the database. Then all the requests can be proceeded locally on the mobile host and the consistency with other database replicas is achieved by anti-entropy [10].

On-demand caching: Normal caching copies objects exclusively on demand, as it is done with the World-Wide Web caches. Rover [5] has an extension of this strategy: it uses migration to import objects into the cache.

Prefetching: This allows to cache in advance some specified data. Prefetching is triggered by explicit application requests or hints. Prefetching is used extensively in Coda [6], or Seer [7]. The former employs a user-defined profile to choose what to prefetch. The latter enables monitoring of file activities in order to measure relationships between files; it computes groups of related files to be prefetched as a single unit.

Mirroring: This approach is widely used in the World-Wide Web. It occurs on explicit request on a whole group of objects, HTML documents for example. In general, World-Wide Web mirroring uses on one hand a single (or group of) source site(s) where the modifications arise, and on the other hand multiple mirror sites where the information is replicated. Mirrors do not accept updates. The replication process can be completely transparent for the holder of the original data. In contrast, the client must redirect its requests to the mirror instead of the original server.

Aggressive caching: This is an extension of on-demand caching. When a caching request occurs, for example on a file, this strategy offers a way to fetch all the directory contents as well. The fetching can be more or less aggressive.

Aggressive prefetching: We consider this strategy as an extension of normal prefetching. As above, the prefetching can occur on group(s) of objects according to the level of aggressiveness chosen. Coda [6] and Seer [7] can be listed in this category.

On-the-fly replication: This strategy allows to transparently fetch information. Fetching occurs on-the-fly according to environment changes, application or users needs. It uses an adaptable grain, replication can happen as well on single objects as groups. In order to determine which objects are essential, both user-defined profile and relationships are used. To achieve the latter, we use object grouping and take benefit of references (i.e. pointers) between objects.

Transparent mirroring: We consider transparent mirroring techniques as a combination of on-the-fly replication and classical mirroring. Mirror creation is still transparent for the holder of the object or group of objects. However the mirror creation is not done on-demand anymore, it is based upon remote object usage, latency monitoring, and so on.

4 Supporting replication

Replication is useful but not sufficient alone. The underlying system has to provide basic mechanisms to handle data, remote use and so on. In Cadmium, we chose the mechanisms listed below. Each of these is needed either for the full replication support, either for adaptation purposes, but for completeness none can be omitted.

Along with objects, we have to deal with references between them [13]. In a mobile environment, the target of a given reference may vary to reflect environment changes, especially if we support replication. To enable an application to refer to the objects it needs, we use a reference redirection protocol we call flexible binding [12, 1]. For instance, a reference to a file would be bound to a dedicated file server when at the office, switched to a cached copy on the road, and then to a secondary server when reconnecting at a new location. The rebinding strategy can be more elaborate and allow to select "nearest" replicas, less loaded servers, take in account the connection state, the available networks, the user or application requests, etc.

Replication also requests mechanisms to manage replica versions, ensure propagation of updates, detection of conflicts and automatic resolution to the extent possible. Along with the replication strategies provided to the applications as a library, we have libraries of strategies for access control, consistency, conflict detection and resolution, filtering, compression and so on.

We intend libraries to provided basic support to the applications. When needed, an application can plug in its own strategies and not use the Cadmium ones anymore. When dealing with strategies, an application can select their scope: a given strategy can be used on a per object, group, or whole application basis. To enable the application to modify its behavior in a highly evolving environment, on-the-fly strategy change is possible. One can for example switch between a strong consistency strategy while fully connected, and a weak consistency strategy while weakly connected or disconnected.

Finally in order to trigger application adaptation, Cadmium has to provide environment monitoring and upcall registration [9] to the applications. These are the resource-awareness hooks. Registrations allow an application to track state changes of critical resources. By instance, it is possible to register thresholds for bandwidth availability. Whenever a threshold is reached, the application gets back an upcall from the system. And as a response, it can chose the appropriate behavior and gracefully handle service degradation (or improvement). Other thresholds can be set upon connection states, available networks, latency, distance from some points, external equipments, etc.

5 Conclusions and perspectives

Cadmium is an on-going research project. We are currently focusing on on-demand caching strategies, migration and reference redirection, as well as on-the-fly replication and reference tracking.

Next steps will deal with adding various strategies to libraries, managing upcalls, tuning strategy selection algorithms, enforcing cooperation between on-the-move mobile hosts, and providing a tentative taxonomy of replication strategies targeted for mobility.

Finally, we plan various measures of our mechanisms such as the responsiveness of our replication strategies according to network availability and bandwidth evolution (soft or burst traffic). We intend upcall registration criteria to highly affect replication efficiency. We will experiment our mechanisms with criteria such as bandwidth thresholds, distance from some specific points or fixed stations, etc. Finally, availability of data and consistency are contradictory. The replacement strategies are supposed to provide the same behavior. We will measure the impact the replacement or consistency strategies on replication efficiency.

References

[1] Baggio, A. System support for transparency and network-aware adaptation in mobile environments. In *ACM Symposium on Applied Computing, special track on Mobile Computing Systems and Applications* (Atlanta, Georgia, USA, Feb. 1998). http://www-sor.inria.fr/~aline/.

[2] Baggio, A., and Shapiro, M. Reconciling transparency with resource awareness in nomadic computing. In *4th Cabernet Radicals Workshop* (Rethimnon, (Crete), Sept. 1997). http://www-sor.inria.fr/~aline/.

[3] Demers, A., Petersen, K., Spreitzer, M., Terry, D., Theimer, M., and Welch, B. The bayou architecture: Support for data sharing among mobile users. In *Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, US, Dec. 1994). http://www.parc.xerox.com/csl/projects/bayou/.

[4] Hinden, R. M. IP Next Generation overview. Tech. Rep. Internet Draft, IETF Mobile IP Group, May 1995. http://www.ietf.org/.

[5] Joseph, A. D., deLespinasse, A. F., Tauber, J. A., Gifford, D. K., and Kaashoek, M. F. Rover: A toolkit for mobile information access. In *Fifteenth Symposium on Operating Systems Principles* (Dec. 1995). http://www.pdos.lcs.mit.edu/rover/.

[6] Kistler, J. J., and Satyanarayanan, M. Disconnected operation in the Coda file system. In *Thirteenth ACM Symposium on Operating Systems Principles* (Asilomar Conference Center, Pacific Grove, US, 1991), vol. 25, pp. 213–225. http://www.cs.cmu.edu/afs/cs/project/coda/Web/docs-coda.html.

[7] Kuenning, G., and Popek, G. J. Automated hoarding for mobile computers. In *Sixteen ACM Symposium on Operating Systems Principles* (Saint Malo, France, Oct. 1997), pp. 264–275. http://fmg-www.cs.ucla.edu/geoff/sosp97.html.

[8] Mummert, L. B., Ebling, R., and Satyanarayanan, M. Exploiting weak connectivity for mobile file access. In *15th ACM Symposium on Operating Systems Principles* (Copper Mountain Resort, Colorado, USA, Dec. 1995). http://www.cs.cmu.edu/afs/cs/project/coda/Web/docs-coda.html.

[9] Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., and Walker, K. R. Agile application-aware adaptation for mobility. In *Sixteen ACM Symposium on Operating Systems Principles* (Saint Malo, France, Oct. 1997), pp. 276–287. http://www.cs.cmu.edu/afs/cs/project/coda/Web/docdir/s16-reprint.ps.Z.

[10] Petersen, K., Spreitzer, M. J., Terry, D. B., Theimer, M. M., and Demers, A. J. Flexible update propagation for weakly consistent replication. In *Sixteen ACM Symposium on Operating Systems Principles* (Saint Malo, France, Oct. 1997). http://www.parc.xerox.com/csl/projects/bayou/.

[11] Satyanarayanan, M., Noble, B., Kumar, P., and Price, M. Application-aware adaptation for mobile computing. In *Sixth ACM SIGOPS European Workshop* (Dagstuhl, Germany, Sept. 1994). http://www.cs.cmu.edu/afs/cs/project/coda/Web/docs-ody.html.

[12] Shapiro, M. A binding protocol for distributed shared objects. In *International Conference on Distributed Computing Systems* (Poznan (Poland), June 1994). http://www-sor.inria.fr/publi/extended-ICDCS.300dpi.html.

[13] Shapiro, M., Dickman, P., and Plainfossé, D. SSP Chains: Robust, distributed references supporting acyclic garbage collection. Tech. Rep. 1799, INRIA, 1992. http://www-sor.inria.fr/publi/SSPC_rr1799.html.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399