Experience With ADSM In Conjunction With AFS And DFS

Stand: 22.10.1999

Since the advent of network based file systems like NFS, AFS, DCE/DFS, or Windows NT, a file can no longer be attributed to a single host or node, and notions like 'file space' loose part of their meaning. Providing a well-defined framework for using ADSM under these circumstances is the more crucial, the more a file systems tends to be world-wide, as are AFS and DCE/DFS. Experience shows that this challenge for ADSM is not yet fully met. - Presentation given at the Oxford University ADSM Symposium 1999.

Who Is Using These Products?

- both widespread in German universities
  - no exact data, tens each, large overlap
- DFS very slowly increasing
  - no dynamics in the product
  - AFS continued
  - little support (e.g. no Linux client)

Both ADSM and either AFS of DCE/DFS are in widespread use in German universities, with a significant number of institutions that use both and are thus interested in full support of distributed file systems by ADSM.

Usage Of AFS At LRZ

- the "general" university staff member
  - mail
  - workstation usage
- department WWW sites
- LRZ's WWW site
- software distribution

Since 1992, AFS has been *the* file system on LRZ's general purpose computers and workstations.

Reasons For AFS Or DFS

- global file system
- distribution of software
- easy administration
- better security
- cooperation across institutions (high performance computing)

Future Use Of AFS Or DFS

- products rather static, in particular DFS
- no alternative now
- emerging new approaches
  - Coda (CMU)
  - GFS (UoMinn)
- DFS becoming a legacy system, even prior to complete deployment?

The question as to whether to switch from AFS to DFS has been a topic of discussion in the LRZ for some years. Initially, there was strong pressure for this transition: AFS was doomed to be discontinued with the larger dissemination of DFS. This has not taken place in a general setting: AFS is still vital, and DFS is extremely slow in getting any momentum. The somewhat provocative term "legacy system" for DFS in this context is meant in the same sense as products like Motif or CDE have meanwhile been overtaken by more flexible solutions such as TCL/Tk, Java, or KDE, which interface more easily with other software and create no licensing problems (where the major problem with licensing is *not* the mere cost of the license but the hassle for the developers when they run into a bureaucracy whenever they want to produce something useful).

This is also how the hint to Coda and GFS is meant. As of now, there is no way that these products could replace DFS. But their mere advent shows that DFS has not been able to meet the requirements of the users of distributed file systems. Formerly, users would then have tried to issue pressure on the vendors of the proprietary products; nowadays, they design new products that are better suited. As the example of Linux shows, such an endeavour can bring about solutions that do not only meet the expectations better but are at the same time at least as reliable as the systems they replace, and due to their higher flexibility and to the absence of licence problems, gain a large market share very quickly.

Usage Of ADSM At LRZ

- 3 different areas
  - campus
  - central (distributed file systems)
  - high performance
- 1200 nodes
  - 60% Unix (15% Sun, 12% Linux, 12% IBM, ...)
  - 40% PCs (20% WinNT, 12% other MS, ...)
- backup: 173 M files (20 TB)
- archive: 47 M files (38 TB)
- traffic: in 5 TB/month, out 143 GB/month

The latest statistic about ADSM usage at the LRZ can be found on a series of Web pages. These are in German, but all you need to know are these words:

gespeicherte Datenmenge	=	*amount of data stored*
Datenverkehr	=	*data traffic*
Anzahl Dateien	=	*number of files*
Ø Dateigröße	=	*average file size*
gesamt	=	*total*

Reasons For ADSM

- best choice
  - strong company
  - large customer base
  - suited for purpose
  - consider: inertia of data
- works fine for campus
- problems with distributed file systems
  - future support repeatedly promised
- suited for high performance?
  - HPSS too expensive (manpower)

ADSM has been chosen as the market leader, and the decision has not been regretted. As far as distributed file systems are concerned, however, ADSM's design is way below our expectations (and below IBM's hype). The main points of dissatisfaction were discussed in 1995 with IBM; they are summarised in a paper which still describes the situation adequately.

ADSM has not been designed for high performance, but at the moment we see no alternative to using it for that purpose as well. HPSS is still in an experimental state and requires prohibitive manpower.

How Are Things Working (Each On Its Own?)

- AFS works fine and stable
- DFS still unstable (tends to crash clients)
- ADSM on campus stable and reliable, but
  - Linux client missing
- ADSM with distributed file systems

The experiences with distributed file systems are addressed in the remaining portion of this lecture.

Backup: Solution With "buta"

- principle: normal built-in AFS or DFS backup, but use ADSM instead of tapes
- product of IBM, but not part of ADSM
- no problems with data model (because it uses AFS's or DFS's data model)
- complete and consistent backup
- no user-driven restore (would not work anyway)
- summary: strongly recommended solution

Two items need an explanation:

The term "data model" has not yet been introduced. Later in this presentation, we will see that one of ADSM's major problems in such environmants is its failure to provide a model in which the data and their relationships can be described.

The parenthetical statement "would not work anyway" refers to the feature of ADSM that in order to do a restoration, a user must be validated not only on the system where the restoration is initiated but also on the system where the backup was taken. In the case of a distributed file system, however, this will often be a dedicated file server with no access by the ordinary user.

Archive: The Permanent Work-Around

- single node name on all hosts
- clients behaviour is undefined
- scalability problems (50 GB database)
- no HSM (would require more flexibility of data model)
- ADSM depends on the fiction of independent hosts

As there is no notion within ADSM to specify that files on different hosts (nodes) are the same, the only way to have access to these files is to use the same node name everywhere. This is dubious from a security standpoint.

Moreover, such a node shared by many users on many hosts tends to have many files which cannot easily be distributed over different servers, thus leading to servers with very large databases. This inhibits scalability as far as it could be achieved by assigning more servers to a given collection of data.

The item "clients behaviour is undefined" alludes to a bug described below.

The remark on HSM pertains to *real* HSM, that is, to the usage of HSM as overflow capacity for the regular filespace on a host. Here, the static nature of ADSM's data model inhibits the usage of ADSM in a rapidly changing environment as we have it in academia. - The usage (or abuse) of HSM as a user interface to an archive, as is done at ECMWF Reading or at Oxford University, does not require the same flexibility of data model.

The Reality Does Not Match The Fiction

- Partitions (volumes, file sets, directories) may be shared among hosts.
- They may have different names on the hosts; these names may change over time.
- The regions where they are visible may overlap.
- Manual "file space" definitions may vary from one host to another.
- They may change over time.
- The authorised usership may be different on hosts sharing the same data.
- Files may be required to survive the host where they resided.

Each of these facts is a problem in the context of ADSM usage. These problems have been summarised in a paper in June 1995, but the points raised then are still valid. The requirements that arise from the rapid changes in such environments are summarised in a requirements specification by the University of Leeds.

But Hosts Do Share Data

- either: have terms to describe the environment
  - "file space" is a first attempt
  - must be able to model future trends

- or: separate name service from data storage
  - major redesign
  - might facilitate true API
- anyway: well-defined interfaces needed
  - lack of these is most prominent problem

Example Of A Problem: Symptoms

- archived file disappears immediately after archival, reproducible
- reason: different file space used for archive and retrieval
- dependent on
  - prior history (file space must have existed before)
  - version of ADSM client (happens since version 3)
- details on http://www.lrz.de/services/datenhaltung/adsm/papers/afsbug/

This is a short description of a problem we encounter at LRZ because the notion of "filespace" has not been clearly defined for environments with distributed file systems.

Example Of A Problem: Background

- commitment vague
- solution vague

On the foils for the lecture, there are excerpts from the documentation and from the text with which the PMR was closed. These texts are also available on the WWW page referenced above.

Problems

- lack of model
  - implicit: collection of independent hosts
  - model could enable modularity
  - HSM impossible without model
- lack (or late delivery) of clients
  - platforms: Linux, Cray, VPP, Hitachi
  - file systems: AFS, DFS (except AIX)
  - too many distinct clients
- lack of well-defined interfaces
- unspecified commitment
- scalability

Here we emphasise again that the specific problem just presented is in no way a single error (and in fact one that - like many others - is being explained away by IBM instead of fixed), but is the necessary outcome of a design which lacks a model of the environment in which the software is to operate. The indispensability of a clear model not only of the behaviour of the software but of the environment in which it is to operate has already been the topic of another lecture on an ADSM meeting.

The Challenge

- cohesive and future-safe object model of all players
- documented interfaces
- modular design
- standardised clients

Trend To Open Source

- independent of "them"
  - swifter support
  - better response to new requirements
- better understanding of architecture
- (much less important:) for free

At first glance, this is something completely different: In the beginning of this lecture, we were talking about the difficulties encountered by proprietary products in a market that demands highly flexible software, whereas quite recently, we talked about the need for a clear specification of interfaces. But there is a connection: the unclear and in some sense proprietary way ADSM specifies its interfaces is exactly what renders such products less competitive than what they could be.

An example is the availability of clients. Had the interface of clients been clearly specified, new clients could emerge at the same pace as they are needed: written by IBM, or by the vendors of the systems that want ADSM clients for their systems, or by third parties. Of course, a prerequisite would be the availability of source code under a licensing agreement that is curtailed for this purpose. Now, getting a new client is a cumbersome and expensive task, and the quality of the product is not always satisfactory (which is not surprising for a product underlying a monopoly).

As has been emphasised before, the main advantages of a somewhat more open approach is not the reduced price, but the swifter and more competent peer-to-peer support, and the greater flexibility to port the product to new environments.

Dreaming Of A More Open ADSM

- Pro:
  - large usership
  - general need for product
  - faster availability of new clients
- Con:
  - intrinsic complexity
- Solution:
  - IBM should learn the lesson why open systems are more successful and combine the best of the two worlds.

ADSM is not a highly specialised product but one that has a large usership with diverging needs, comparable to an office suite, a graphics package, or an operating system. Typically, these are the kind of products where open-source products with outstanding quality emerge. ADSM development should take that as an incentive to learn from the open-source world what makes a product suitable for the market demands of today.

Complexity Is The Main Issue

- complexity of the system
  - missing or outdated functional model
  - evolutionary development
- critical complexity reached
  - file space mapping
  - database resilience
- Conclusion: Each necessary enhancement must reflect a simplification of the underlying model.
  - evolutionary redesign
- Make it as simple as possible - but no simpler (Albert Einstein)

This foil has been stolen from a lecture about reliability, also pertaining to ADSM, in order to show that devising a functional model is not only a requirement for correct functionality but also for reliability.

The Message

- ADSM was not designed with distributed file systems of any type in mind.
- Their inclusion cannot be done as a simple add-on; rather, a consistent data model has to be developed and documented.
- Everything else is a collection of dubious work-arounds, which admittedly may work for some time.

Impressum, Helmut Richter, 22.10.1999, letzte Änderungen