Advanced Accounting for AIX 5L Version 5.3 White Paper

By Luke Browning

July 27, 2004

Abstract

AIX 5L™ V5.3 is introducing a new mainframe-inspired accounting subsystem containing many new advanced features like interval accounting, data aggregation, system administrator defined billable entities and dynamic classification of accounting data. The name of this new subsystem is Advanced Accounting for AIX 5L V5.3.

Introduction

Increasingly, businesses are treating their data centers as cost centers rather than overhead expense and charging cost back to their user communities based upon the amount of computing services they use. Chargeback is an effective means of managing IT (Information Technology) costs, because it places the cost burden on the end user of the computing services. Within most organization, it is typically at the level of the end user (i.e. individual, department, or project) where budgets exist and costs are carefully managed.

Chargeback involves two fundamental components, consisting of accounting data and report generators. Accounting data is needed that is sufficient to support cost analysis and billing. From a cost perspective, the data must account for units of work that are relevant to the workload, system environment, and the billing strategy. Accounting data maps units of work to resource utilization, so that the cost of a given work unit can be evaluated. This information is needed for chargeback, so that clients can be charged based on the resources that they use. Report generators are applications that simply generate billing reports from accounting data. In general, these applications needed to be highly customized for each organization's business environment and are provided by clients and ISVs.

Most UNIX® operating systems (OS), including AIX 5L, support accounting mechanisms that provide accounting data and facilities for the purposes of chargeback. However, these mechanisms are primarily geared around traditional UNIX time-share environments, where the units of work to be accounted for are UNIX processes and users are system-defined entities. The mechanisms have clearly not kept pace with the expanded use of UNIX OS beyond time-sharing and changes in UNIX workload and system environments. They do not considered key units of work, such as logical partitions (LPARs), workloads and application transactions that have become relevant to chargeback in many end user environments or supportive of IBM technical strategies (i.e. LPAR, NAS). Additionally, these mechanisms do not consider user entities that are meaningful to chargeback to many end user environments such as application-defined users or organization-defined users such as departments, divisions or projects. Finally, the existing mechanism offers facilities for managing the accounting environment that are relatively inflexible and non-scalable, making it difficult for clients to use these mechanisms in their chargeback schemes.

A new mechanism is needed to address the shortcoming of the existing mechanisms and the expanded use of UNIX systems. Under this new mechanism, accounting data needs to be provided for key units of work and users that are important for chargeback.

Work Units

There are four different units of work scenarios of interest, consisting of processes, workloads, LPARs and application transactions.

Processes. Advanced Accounting supports process level accounting consistent with today's accounting mechanism. This allows current users of the existing mechanisms to move to advance accounting and benefit from the improved facilities that are offered under the advance accounting framework (i.e. interval accounting)

Workloads. Within a single operating system image, there is a need to classify processes into workload-related classes and generate accounting data at the class level. This allows accounting lines to be aligned with workload boundaries and is needed for environments where multiple process-based workloads have been consolidated within a single OS image. Classification of processes is supported without requiring changes to application or middleware and is automated based upon system administrator-defined class assignment rules.

Logical Partitions. Accounting data must be provided that reflects gross level resource utilization for LPARs to allow chargeback of a single partition (i.e. a single partition as a work unit). To satisfy this requirement, Advanced Accounting provides accounting data for system resources like processors, memory, disks, network interfaces, file systems, and virtual devices.

Application Transactions. When applications process transactions for multiple accounting lines, the previously mentioned forms of accounting cannot be used to perform chargeback. In such cases, only middleware and applications are able to identify work units for accounting lines. Examples of transactions may be a data base operation or WebSphere® application. In the networked world, the ability to chargeback transactions is critical, because work occurs within this environment as transactions flow through systems across networks.

Users

In an eUtilities-based system, users are entities and not what is considered a user in a traditional time share system where operating system-defined users are meaningful. A user may be any entity that is of interest for chargeback, like a person, department, division or project and can vary from installation to installation. The key requirement here is to allow these installation-defined user entities to be mapped to units of work. This requirement is satisfied by tagging individual work units with project identifiers.

In general, user entities are represented and uniquely identified through an installation's business data, such as an employee number, department number, or project accounting number. Work unit tagging enables an installation to tag work units with an externally specified project identifier. Tags are not interpreted by AIX 5L (and by middleware) and may be specified such that they contain user entity ids. When tagged, all accounting data records constructed for the work unit contain the project identifier. This enables the user of accounting data to map a work unit to a user entity.

Target Markets

Advanced Accounting plays in a number of major spaces. To begin, it is critical to service providers, like IGS (IBM Global Services) and EDS, who are fundamentally in the business of charging for computing services. Service providers require the ability to accurately determine the use of computer resources at many levels in order to offer competitive and flexible chargeback models to their diverse client sets and support their own visions of on demand computing. LPAR, process, workload and application transaction accounting are all important for this environment.

Advanced Accounting is also important to enterprise data centers. The chargeback of computing services to the users of these services has long been the practice in most large client data centers. The IT organizations that manage these data centers desire the ability to provide chargeback schemes that are consistent with their business and workload environments. Key accounting requirements for this environment are project codes and system resource accounting.

The high performance computing (HPC) environment has a strong need for Advanced Accounting. Most HPC shops perform chargeback of computing services. Key requirements for this environment are LPAR accounting, HPC project codes and system resource accounting.

Advanced Accounting is a critical piece of several key AIX 5L strategies. It is a basic building block for Grid Computing and On-Demand computing, because it provides the metrics that these computing environments need to enable chargeback.

Advanced Accounting Features

The new accounting framework has the following features:

- Report and Analysis Tools
- Projects and Project Assignment Policies
- Interval Accounting
- Improved Statistics and Selectable Accounting Records
- Data Aggregation
- Scalable and Robust I/O
- Projects and Dynamic Classification

- Co-existence with UNIX Accounting Utilities
- Application Transactions

Report and Analysis Tools

These tools are highly customized for each client's business environment. They are not provided by AIX 5L. They are provided by clients and ISVs.

AIX 5L provides the following sample command **readaacct** to format accounting data files.

/usr/sample/bin/readacct [–f file] [–t <TRID>] [-b <begin_time>] [-e <end_time>] [-c] [-h]

The –f parameter names the input data file.

The –t parameter identifies the Transaction IDs (TRIDs) that should be displayed. TRIDs produced by AIX 5L are defined in the include file <sys/aacct.h>.

The –b and –e parameters are used to limit the search by time. The format for the time parameter is "mmddHHMM", where mm is month, dd is day, HH is hour using the 24-hour clock, and MM is minute.

The –c flag may be used to produce a colon separated view of the data. It can be used to import accounting data into spread sheets and databases, where the data can be further processed to produce client bills.

The –h flag is used to display the accounting file header, which contains information about the file and the system that collected the data including the system id, model number, hostname, LPAR name and partition number.

The format of accounting data files is documented in the header file <sys/aacct.h>. The general format of the file can also be seen in a source code example. The source code for the sample command **readaacct** command is located in the same directory as the command.

Projects and Project Assignment Policies

Project codes are system administrator defined tags that are associated with work units via system administrator defined project assignment policies. Advanced Accounting produces an accounting record for each work units. Each accounting record identifies the project code that was assigned to the work unit by the assignment policy.

The billable account is named by the project code. Conceptually, project codes are indices into end user account information, which identifies the mailing address, account number, service level agreement, discounts, etc. of the client. The billing application is

expected to add up all of the accounting records with the same project code, when produce a bill for that account.

System administrators define assignment policies to control the assignment of projects to work units. A policy is typically composed of multiple assignment rules, which are downloaded into the kernel and used to control manual and automatic project assignment.

Multiple project assignment policies are supported. They are:

- Admin Policy
- User Policy
- Group Policy

The Admin Policy is similar to a AIX® Workload Management (WLM) policy. It uses process attributes to associate project identifiers with processes. These attributes are User id, Group id and the pathname of the executable. The Admin Policy uniquely provides the ability to classify applications, which is interesting from an accounting perspective, since it enables the use of "premium" applications to be flagged on an individual billable account basis.

Multiple Admin Policies may be specified enabling time-based accounting policies to be implemented. Administrators may define separate policies for peak and off-peak charges. The **cron** facility may be used to load policies at the appropriate time.

Assignment rules may also be specified at the user and group level through the commands **mkuse**r, **chuser, mkgroup,** and **chgroup.** Assignment rules specified in this way are collectively referred to as User and Group Policies. The motivation for these policies is that some sites have thousands of users and it is more important to provide basic functionality in a well integrated fashion that does not require additional administrative steps. Accounting policies may be specified directly through the basic user administration commands.

The Admin, User, and Group Policies may be active at the same time. Internally, loaded policies are referenced sequentially in the following order: Admin, User, and Group. Policies have to be loaded to take effect. It is not sufficient to create them.

Interval Accounting

Interval accounting enables accounting records to be produced and collected at system administrator defined intervals. Separate intervals are defined for the collection of process and system resource related data.

The process interval may be used to generate intermediate process records enabling long running jobs to be factored into current bill. This is an important advancement, because most UNIX accounting systems do not have the capability to produce intermediate records, which introduces delays into the billing system. The billing system must wait

for a job to complete before an accounting record is produced. Some jobs like data modeling applications can run for months, so this can be a significant factor to the bottom line of a business in some cases. These delays can be eliminated with intermediate process records and interval accounting.

The system interval collects accounting information related to the system level use of logical and physical resources like processors, memory, disks, network interfaces, virtual adapters and devices, and file systems. The information produced by the system interval is used for LPAR level accounting. This data has other uses as well. It may be used by capacity planning and performance analysis tools, since it provides a mechanism to profile the use of system resources.

For example, it provides a historical view of file system and disk utilization, so it can be used to identify the specific file systems that were busy at some point in the past. This information can be cross referenced with disk utilizations to ensure that files systems are properly balanced over disks.

It is recommended that the system interval be enabled to run once an hour.

It is recommended that the process interval be enabled to run once a day.

Enhanced Statistics and Selectable Accounting Records

Advanced Accounting provides enhanced process statistics:

- Process and total thread elapsed times
- Microsecond level granularity in dispatch times
- Elapsed page seconds of memory usage
- Local and remote logical file I/O
- Local and remote socket I/O

Note that the process record provides an indication of network usage.

The accounting data file has a flexible file format and can accommodate different types of records. The following accounting records are currently supported:

- Process record (Transaction Identifier 1)
- Aggregated process record (TRID 2)
- Aggregated application record (TRID 3)
- System CPU and memory (TRID 4)
- Project assignment policy record (TRID 5)
- File system record (TRID 6)
- Network interfaces (TRID 7)
- Disk record (TRID 8)
- Lost accounting record (TRID 9)

- Server virtual device record (TRID 10)
- Client virtual device record (TRID 11)
- Third party kernel extension aggregation record (TRID 12)
- ARM application environment record (TRID 13)
- ARM transaction environment record (TRID 14)
- ARM transaction instance record (TRID 15)
- ARM aggregated transaction instance record (TRID 16)
- Project definition record (TRID 17)

The administrator can enable or disable most accounting records individually. The accounting records with the following TRIDs: 2, 3, 13, 14, and 16 cannot be disabled. To disable process accounting, it is necessary to disabled TRID 1. To disabled the collection of ARM data, it is necessary to disable TRID 15.

Selectable records minimize the amount of accounting data that is produced and is an important tool for tuning the performance of Advanced Accounting. Not only does it minimize the amount of I/O that needs to be performed, but in many cases it eliminates additional accounting overhead such as the incrementing of internal statistics.

Data Aggregation

Data aggregation adds accounting records of the same type in memory, so that fewer accounting records are written to the accounting data file. Data aggregation can be used to reduce the I/O requirements of Advanced Accounting. It is one of the tools that can be used to tune the performance of the Advanced Accounting subsystem. It is intended to be used in conjunction with application transactions, which can theoretically produce accounting data at the millisecond level.

The system knows how to intelligently aggregate accounting data that it produces, so that almost nothing is lost. For example, the aggregated process record lacks only a couple of fields in the process record. Namely, the PID, GID, and WLM class. It doesn't make sense to aggregate these fields, since the goal is to minimize the number of records that are produced. In general, the aggregated record contains sufficient information including the project code and resource utilization that is needed to perform chargeback. However, it is a different record, so it is necessary to ensure that the billing application supports the aggregated form of the record.

The converse is true for vendor data. The operating system does not know the format of vendor accounting data, so it can't intelligently aggregate it. To compensate, it keeps statistics about the occurrence of the vendor data and the aggregate resource utilization, which it reports in lieu of the vendor data. This information is reported with the project code, so that the billing application can relate the aggregate resource utilization with a project code. The format of this data can be seen by examining the relevant structures in the header file <sys/aacct.h>. The structures **tran_agg_KE** and **tran_agg_arm** are associated respectively with TRIDs 12 and 16.

Interval accounting is used to periodically write aggregated accounting records to the accounting data file, so it must be enabled if data aggregation is enabled. The process interval is used to commit aggregated process accounting records (TRIDs 2 and 3). The system interval is used to commit aggregated third party kernel extension records and ARM records (TRIDs 12 and 16). The process interval should be set to once an hour, if the aggregation is enabled for process accounting.

Scalable and Robust I/O

The following items are included to ensure that Advanced Accounting scales well.

- Kernel buffering of accounting data
- Asynchronous I/O of kernel buffers
- Pre-allocated files to ensure disk blocks are contiguous for raw device performance
- Selectable transactions only produce the required data
- Data aggregation

None of these features are supported by UNIX Accounting Utilities.

Co-Existence with UNIX Accounting Utilities

Advanced Accounting is implemented independently of UNIX Accounting Utilities. Both may be run at the same time.

Application Transactions

Application and middleware support is required to provide transactions as units of work for chargeback purposes. This is accomplished through the use of the ARM APIs which provide a general framework for describing and managing transactions. Advanced Accounting works with the ARM APIs to capture information about transactions in the accounting data file, so that it can be post processed for chargeback purposes.

The following document describes the ARM APIs.

Application Response Measurement (ARM) Issue 4.0 – C binding
ISBN: 1-931624-35-6
The Open Group
ogspecs@opengroup.org

This document is available at www.opengroup.org/tech/management/arm

IBM Corporation
Systems and Technology Group
Route 100
Somers, New York 10589

Produced in the United States of America
July 2004