



166166

**META DATA PROCESSOR FOR CONVERTING PERFORMANCE
DATA INTO A GENERIC FORMAT**

5 This invention relates to a device and method
for evaluating computing capacity for institutions that
employ multiple computers. More particularly, this
invention relates to a generic data processing device
and method for converting computer system performance
data from a first format to a second format.

10 Companies that own and operate computers for
data processing encounter a need for capacity planning
of computing resources, so that they can efficiently
and accurately plan the purchasing of new computing
resources. Computing resources include CPUs, memory,
15 disk storage, tape storage, access devices, operating
systems, file systems, and many others. Capacity
planning relies on the accurate forecasting of resource
utilization. Forecasting, in turn, requires analysis
of current and historical system performance metrics
20 data. These metrics include CPU utilization, disk
storage utilization, memory utilization, memory
allocation, file system access, and many others.

25 There are several issues of concern with
regard to capacity planning. It is important for
companies to be able to determine points at which new
hardware will become necessary to meet system
requirements. It is also important for companies to be
able to project scenarios for potential configuration
changes including both hardware and software. Another
30 issue of concern is the monitoring and analysis of
performance problems.

To address these and other needs, data
analysis/reporting tools for analyzing, reporting, and
graphing system performance data for the purposes of



G00002

capacity forecasting and planning is currently commercially available. One such product that is widely used is SAS IT Service Vision software available from the SAS Institute, Inc. of Cary, North Carolina. However, performance data must be provided to SAS IT Service Vision in properly formatted SAS datasets. Likewise, specially formatted performance data is required by other commercially available data analysis software.

There are software products available, known as collection agents, that run on computers and collect raw performance data from computer resources. Examples of collection agents include Patrol available from BMC Corporation of Houston, Texas; Unicenter TNG available from Computer Associates of Islandia, NY, BGS available from BMC Corporation, and Candle Availability Command Center from Candle Corporation. Most of the available collection agents may compile performance data into flat files known as Universal/Uniform Data Format (hereinafter UDF) files. A significant problem with available collection agents is the UDF files they produce are not properly formatted for use by data analysis/reporting tools such as SAS IT Service Vision. Furthermore, different types of collection agents may compile UDF files having different arrangements, using different variables and sequential ordering of variables. Data from the UDF files must be appropriately processed to produce properly formatted datasets that may be read and used by data analysis/reporting tools.

Heretofore, it has been necessary to process data from each type of collection agent in a unique way to produce properly formatted datasets. Often, a customized data processing program had to be written

for each collection agent. Further complicating this task is the fact that a single UDF file contains data for many different types of performance metrics; these data must be sorted out into individual dataset tables for input to data analysis/reporting tools such as SAS IT Service Vision.

Accordingly, there is a need for a single integrated product that can read performance data from many different types of collection agents and convert that performance data into properly formatted SAS datasets for use by data analysis/reporting tools irrespective of the type of collection agent that produced the performance data.

The present invention is a data processor for processing performance metric data. The invention functions as a generic interface that facilitates communication between any one of a number of collection agents and data analysis/reporting tools.

In accordance with the present invention, computer system metrics performance data contained in UDF files is converted into SAS datasets for input to data analysis/reporting tools such as SAS IT Service Vision.

The present invention may also perform data processing by identifying the source of the UDF file received, transposing the UDF file data into properly formatted records, sorting the records into individual tables in accordance with the type of metrics reported, and providing the dataset to the data analysis/reporting tool.

The present invention is further directed to an apparatus for converting performance metric data from UDF files to a format readable by data analysis/reporting tools such as SAS IT Service Vision.

The apparatus includes a data processor programmed to retrieve UDF files from various collection agents that may be resident on computers. Each UDF file includes several records that contain performance metric data for the computers and the performance metric data is separated into various fields. The data processor is further programmed to reformat the UDF files, map each UDF file to a dataset and sort the dataset by grouping the records/observations of the dataset by field/attribute. The data processor is additionally programmed to build performance data tables using the sorted dataset.

Still further, the present invention is directed to a system for processing and analyzing performance metric data. The system includes a number of nodes, e.g., computers, where each computer has a collection agent. The collection agents collect performance metric data from the computers and compile the performance metric data into UDF files. A data analysis computer is provided that retrieves UDF files from the collection agents and selects those UDF files produced by the same type of collection agent. The data analysis computer then reformats the selected UDF files and maps the reformatted files to a dataset. The dataset is then sorted by grouping the records/observations by field/attribute, unique datetime stamp, and performance data tables are built by the data analysis computer using the sorted dataset. The performance data tables are in this case SAS datasets, however, Oracle, Sybase, Informix, DB2, SQLServer, or any other database product may be used.

An advantage of the present invention is that it provides a single, generic handshake interface between different collection agents and data

analysis/reporting tools. The invention is not proprietary and does not rely on any specific vendor's collection agent or computer hardware. Using the present invention, companies can efficiently provide services such as capacity planning and other forecasting and diagnostic services to their customers who may use different collection agents. This reduces the burden placed on customers to obtain an interface that will appropriately process performance metric data collected by the customer's collection agent. In addition, it simplifies the task of the service provider and allows it to confidently market its services to a wide variety of customers irrespective of the type of collection agent or computer hardware the customers employ.

A more specific advantage of the invention is that by processing the performance metric data into SAS datasets, the volume of performance metric data input to the data analysis product is reduced by as much as a factor of four from prior art systems.

FIG. 1 illustrates a block diagram of a system architecture in accordance with the invention.

FIG. 2 depicts a comma-delimited UDF file produced by a collection agent.

FIG. 3 shows process architecture for a generic data processor in accordance with the invention.

FIG. 4 is a flowchart illustrating the operation of a first program module in accordance with the invention.

FIG. 5 is a flowchart illustrating the operation of a second program module in accordance with the invention.

FIG. 6 is a flowchart illustrating the operation of a third program module in accordance with the invention.

FIGS. 7A, 7B and 7C represent the code for performing the process depicted in FIG. 4.

FIGS. 8A, 8B, 8C, 8D, 8E, 8F, 8G, 8H, 8I, 8J, 8K, 8L, 8M, 8N, 8O, 8P, 8Q, 8R, 8S, 8T, 8U, 8V and 8W represent the code for performing the process illustrated in FIGS. 5 and 6.

FIG. 9 shows an exemplary computer system for use in the present invention.

FIGS. 10A, 10B, 10C and 10D depict UNIX system metrics.

The present invention is a generic data processor for converting data, particularly performance metric data, into a form readable by data analysis/reporting tools such as SAS IT Service Vision. Figure 1 illustrates a system in accordance with a preferred embodiment of the present invention. In the system, capacity forecasting and planning services are provided for the computing resources embodied by a plurality of nodes 10. In the preferred embodiment, the nodes 10 are computers such as UNIX, Windows NT and PC based workstations. A proprietary collection agent 15 runs on each node 10. Collection agents 15 collect raw performance data or performance metric data from the computer resources. These data include CPU utilization, memory utilization and allocation, storage device utilization and allocation, and other types of metrics. Exemplary UNIX system metrics are depicted in Figures 10A-10D. Examples of collection agents 15 that may be used in the system include the aforementioned Patrol, Unicenter TNG, Candle, and BGS.

Collection agents 15 compile the raw performance data and writes them into flat files known as Universal/Uniform Data Format (UDF) files. A single UDF file may contain performance data for many different types of metrics. The UDF files produced by collection agents 15 are similar in that they include data values for system performance metrics, along with data identifying the source of the metrics. In addition, the data values are arranged in character delimited records, preferably linear comma or linear tab delimited records. However, the UDF files may differ in the specific variables used and the sequential ordering of variables. As used herein, the term "variable" refers to specific performance values of the records.

An example of a comma-delimited UDF file produced by a collection agent is depicted in Figure 2. This particular example was produced by BMC's Patrol. Each line represents a record of a metric collected. Each record comprises the following fields:

Node -- computer/processor/machine from which the metric was taken or to which the metric is attributed; in the first line in the Figure 2 example, node="normet09".

Application -- application or resource from which the metric was taken or to which the metric is attributed; in the first line in the Figure 2 example, application = "DISK". Other examples

are CPU, File System, memory,
network, etc.

Instance -- identifies specific instance of
5 resource (application) that the
metric came from; in the first line
in the Figure 2 example, instance =
"cl Otl 00", which identifies a
specific DISK.

10
Parameter -- type of metric collected (i.e., CPU
utilization) for the specified
application; in the first line in the
Figure 2 example, parameter =
15 "DSKMsp".

Date -- date the metric was collected; in the
first line in the Figure 2 example,
date = "1997-12-01".

20
Time -- system timestamp of when the metric
was collected; in the first line in
the Figure 2 example, time =
25 "06:07:23". In Figure 2, date and
time are actually one field, since no
comma delimits them. Parsing can
separate them into two fields.

Metric -- the actual metric, representing the
30 payload data; in the first line in
the Figure 2 example, metric = 1".

An input UDF file typically comprises these or similar fields, although these fields may be arranged in different orders and have different formats, depending on the specific collection agent.

5 The data contained in the fields may be in a flat file, with records delimited by characters, e.g., commas, tabs, spaces, or the like.

The UDF files are transmitted to a data analysis computer 20 for processing. The UDF files are processed so they may be read by the data analysis/reporting tool 30. In the preferred embodiment, the data analysis computer 20 is a UNIX Midrange Server. However, the data analysis computer 20 may be a mainframe, an NT server, PC's, phone

10 system, fax machine, or any other machine capable of storing or writing performance data. Preferably, the UDF files are transmitted via File Transfer Protocol (FTP) over an Internet Protocol (IP) WAN. However, other methods of transferring files may be used.

In keeping with an aspect of the invention, a generic data processor 25 is operatively engaged with the data analysis computer 20. The generic data processor 25 receives each UDF file, reads the first line of the UDF file and determines the specific arrangement of data contained therein. The type of collection agent 15 that produced the data may be determined from the arrangement of the data. The generic data processor 25 comprises program code that identifies the UDF formats of prespecified versions and types of commercially available collection agents 15. The generic data processor 25 then transposes and reformats the performance metric data and sorts the reformatted performance metric data into individual tables referred to as Performance Data

15
20
25
30

Tables for each type of metric reported. The resulting Performance Data Tables are suitable for input into data analysis/reporting tool 30. Preferably the Performance Data Tables are in the form of SAS datasets.

The SAS datasets are then input to the data analysis/reporting tool 30, in the preferred embodiment SAS IT Service Vision. The SAS IT Service Vision integrates the performance data into daily, weekly, yearly, etc., groups of data; stores data in a Performance Database 35 (PDB); and produces graphical displays of performance data that are useful for capacity forecasting and planning. Reports and data views from the PDB may be retrieved by any known method. A PC 40 running Desktop SAS can access it directly over a LAN/WAN, or a Web Server 45 can provide an interface over an IP network for PCs with Web Browsers such as PC 50.

Turning now to more specific operational aspects of the invention, Figure 3 illustrates the process architecture of the generic data processor 25. Generic data processor 25 is preferably comprised of two SAS programs resident on the data analysis computer 20. The first program, referred to herein as BLDSASDS, receives UDF input files 55, parses the data contained in the files, and builds an SAS output dataset 60 by mapping input data fields to specific attributes of the output SAS dataset 60. Other attributes are derived. These attributes are:

NODE	mapped from node field in input UDF file
APP	mapped from application field in input UDF file

INSTANCE mapped from instance field in input UDF file
PARM mapped from parameter field in input UDF file
DATE mapped from date field in input UDF file
TIME mapped from time field in input UDF file
5 HOUR derived from time field in input UDF file
METRIC mapped from metric field in input UDF file
DATETIME derived by concatenating date and time fields
in input UDF file
QUARTER derived by dividing hour into four segments
10 ZONE derived from date and hour fields; represents
shift during the week. Three zones are
defined.

15 It should be noted that records of UDF files correspond
to observations of SAS datasets. In addition, fields
of UDF files correspond to attributes of SAS datasets.

The first program BLDSASDS produces one
output SAS dataset 60 for multiple UDF input files 55;
the multiple UDF files 55 that result in a single
20 output SAS dataset 60 are produced by the same type of
collection agent (e.g., BMC Patrol), but represent
different time segments and can come from multiple
nodes (computers). In the preferred embodiment,
BLDSASDS collects all UDF files for a single date from
25 a single type of collection agent, and produces a
single SAS dataset.

The second program, referred to herein as
META PROCESSOR, takes the output SAS dataset 60 built
by BLDSASDS and produces multiple Performance Data
30 Tables 65. In the preferred embodiment the Performance
Data Tables 65 are in the form of an SAS dataset that
is formatted for SAS IT Service Vision. Further, a
Performance Data Table 65 is built for each "APP"
attribute value.

A flowchart for BLDSASDS is shown in Figure 4. Figure 7 depicts the SAS code for BLDSASDS. In step 110, BLDSASDS reads multiple UDF input files 55 from a specified directory on the data analysis computer 20 on which BLDSASDS runs. Based on the specific arrangement of data, BLDSASDS identifies the collection agent 15 that produced each UDF file. The first program, BLDSASDS then retrieves multiple UDF input files that are produced by the same type of collection agent 15. Preferably, these multiple UDF files 55 represent a single day's data that have been collected by the same type of collection agent (e.g., BMC Patrol) for multiple nodes.

In step 120, BLDSASDS reads the textual contents of each UDF file 55 produced by the same type of collection agent 15 and builds an output SAS dataset 60 from these contents. Each line of the UDF file 55 is parsed into input fields, based on comma-delimiters or other pre-programmed rules. Each input field of a record in the UDF file 55 is then mapped to a specific attribute of a record in the output SAS dataset 60 (SAS records are referred to as observations). Other attributes are then calculated, as described in reference to Figure 3.

In step 130, BLDSASDS sorts the observations in the output SAS dataset 60 by specific attribute sort order, i.e., by "NODE", "APP", "INSTANCE", "DATE", and "HOUR". BLDSASDS also removes duplicate observations.

In step 140, BLDSASDS names the output SAS dataset 60 for the date on which the metrics were collected. The dataset 60 is then ready for the META PROCESSOR.

Figure 5 is a flowchart illustrating the process performed by the META PROCESSOR. Figure 8

depicts the SAS code for the META PROCESSOR. In step 210, the META PROCESSOR reads the output SAS dataset 60 produced by BLDSASDS. In step 220, the META PROCESSOR groups observations that have the same "APP" field into discrete tables. The output of the META PROCESSOR is multiple Performance Data Tables 65, with each table containing performance data for a single "APP".

In step 230, the META PROCESSOR builds a Performance Data Table 65 for each value of "APP". For example, a Performance Data Table 65 is built for metrics on CPU, disk, file system, kernel, memory, network, NFS, Oracle, Patrol Agent, total Processes, Active Processes, SMP, SWAP, Sybase, and User. As shown in the META PROCESSOR source code in Figure 8, a process may be performed for each "APP" Performance Data Table 65. These processes are similar.

Figure 6 is a flowchart illustrating the sub-process of step 230 of Figure 5. By way of example, Figure 6 is directed to building a Performance Data Table for the File System metric. However, as illustrated in Figure 8, similar processes may be employed for building Performance Data Tables 65 for other metrics. Step 310 indicates that this process is performed on each observation of the Performance Data Table 65. In step 220 of Figure 5, all observations for APP = FILESYSTEM are grouped into a distinct table. In the sub-process of Figure 6, each observation in this table is acted upon, and is referred to as the input observation. Output observations are created by the META PROCESSOR for the Performance Data Table dataset.

In step 320 an observation for the output Performance Data Table dataset is built by keeping static attributes from the input dataset; these

attributes include "APP", "DATE", "DATETIME", "HOUR",
"INSTANCE", "NODE", "PARM", "QUARTER", and "TIME".
Three additional attributes are created: "FSCAPCTY",
"FSFINODE", and "FSINPCTU". These represent the three
5 types of metrics collected for the File System
application. Other applications (values of "APP"
attribute) will have different numbers of metric
types collected. The metric type collected is
indicated in the "PARM" attribute.

10 In the next three steps, the value of the
"METRIC" attribute in the input observation is assigned
to one of the three attributes created in step 320, in
accordance with the value of the "PARM" attribute in
the input observation. More particularly, in step 330,
15 if the value of the "PARM" attribute of the input
observation is equal to "FSCapacity", then the value of
the "METRIC" attribute is assigned to "FSCAPCTY" in the
output observation in step 340. Here, "FSCAPACITY"
represents the capacity of the file system. If the
20 value of the "PARM" attribute does not equal
"FSCapacity" then the process proceeds to
step 350.

In step 350, if the value of the "PARM"
attribute of the input observation is equal to
25 "FSFreeInodes", then the value of the "METRIC"
attribute is assigned to "FSFINODE" in the output
observation in step 360. Here, "FSFINODE" represents
the number of free I-nodes. If the value of the "PARM"
attribute does not equal "FS FreeInodes" then the
30 process proceeds to step 370.

In step 370, if the value of the "PARM"
attribute of the input observation is equal to
"FSInodeUsedPercent", then the value of the "METRIC"
attribute is assigned to "FSINPCTU" in the output

observation in step 380. Here, "FSINPCTU" represents the number of free I-nodes. If the value of the "PARM" attribute does not equal "FSInodeUsedPercent" then the process proceeds to step 390.

5 Step 390 is an optional error processing step that may be performed when the "PARM" attribute of the input observation does not equal any of the values set forth in steps 330, 350 or 370. This error processing step may include defaulting the output observation to a
10 pre-defined value or string, and continuing with the next observation.

In step 400, a check is performed to determine whether all observations have been processed. If they have not, then the process returns to step 320.
15 If they have, then step 410 is performed whereby the processed observations are summarized by NODE, APP, INSTANCE, DATETIME, DATE, TIME, and HOUR. The META PROCESSOR then outputs the Performance Data Table in step 420.

20 It should be noted that all Performance Data Tables built by the META PROCESSOR in step 230 of Figure 5 are provided to the data analysis/reporting tool 30 in an appropriate format for further processing. The data analysis/reporting tool 30 may
25 then produce reports and graphs for a display containing a variety of system information that is of significant value to Information Technology professionals.

30 In the preferred embodiment of the present invention, the nodes 10 are UNIX-based midrange computers, such as DEC Alpha servers and IBM RS/6000 computers. However, in alternate embodiments, the nodes 10 may comprise other computers using different operating systems and hardware configurations. In

further alternate embodiments, the nodes 10 may comprise other information devices including networks, phone systems, fax machines, or other devices capable of storing or writing performance related data.

5 The data analysis computer 20 employed in the preferred embodiment is an IBM AIX RS 6000 UNIX server illustrated in Figure 9. An EMC disk array (not shown) having 270GB of storage may be attached to the server. The data analysis computer 20 is not limited to a UNIX
10 system. The data analysis computer 20 could be a PC, a mainframe, Windows NT workstation, or any other computing device.

 The data analysis computer 20 includes one or more processors 605. Processor 605 is connected to
15 command bus 610. The data analysis computer 20 may communicate with other systems such as PC 40 or web server 45 via a network 615.

 Data analysis computer 20 also includes a main memory 620, preferably random access memory (RAM),
20 and a secondary memory 625. Secondary memory 625 includes, for example, a hard disk drive 630, a PROM 635 and/or a removable storage drive 640, representing a floppy disk drive, magnetic tape drive, a compact disk drive, etc. Removable storage drive 640 reads
25 from and/or writes to a removable storage unit 645 in a well known manner.

 Removable storage unit 645, also called a program storage device or a computer program product, represents a floppy disk, magnetic tape, compact disk,
30 etc. As will be appreciated, removable storage unit 645 includes a computer usable storage medium having stored therein computer software and/or data.

 Computer programs (also called computer control logic), such as BLDSASDS and the META PROCESSOR

are stored in main memory and/or secondary memory 625. Such computer programs, when executed, enable data analysis computer 20 to perform the features of the present invention as discussed herein. In particular, the computer programs, when executed, enable generic data processor 25 to perform significant features of the present invention. Accordingly, such computer programs represent controllers of the data analysis computer 20.

In an alternate embodiment, the invention is directed to a computer program product comprising a computer readable medium having control logic (computer software) stored therein. The control logic, when executed by the generic data processor 25, causes the generic data processor 25 to perform the functions described herein.

While various embodiments of the present invention have been described, it should be understood that they have been presented by way of example only, and not limitation. While the present invention is particularly suited to function as an interface between available collection agents and SAS IT Service Vision software, it is not limited to this function. The invention may be used to convert performance metric data from a variety of collection agents to datasets. These datasets may be used by SAS IT Service Vision or any other appropriate data analysis/reporting tool. Further, any type of performance data can be processed. Metrics collected by database collection agents and network collection agents may be used as well as those collected by UNIX collection agents. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary

embodiments, but should be defined only in accordance with the following claims and their equivalents.

We claim:

1. In a computer, a method for converting performance metric data, produced by a plurality of types of collection agents resident on a plurality of nodes, from UDF files to a second format, said method comprising:

retrieving a plurality of UDF files from a corresponding plurality of collection agents, each UDF file including a plurality of records containing performance metric data corresponding to one of the plurality of nodes, the performance metric data being separated into a plurality of fields;

selecting UDF files produced by one of the plurality of collection agents;

reformatting each UDF file and mapping each UDF file to a first dataset;

sorting the first dataset including grouping the records by attribute; and

building a plurality of performance data tables using the sorted first dataset.

2. The method for converting performance metric data of claim 1 wherein reformatting each UDF file includes parsing the data fields of each record and mapping the parsed data fields to a plurality of attributes.

3. The method for converting performance metric data of claim 2 wherein said selection step precedes said retrieving step.

4. The method of claim 2 wherein said retrieving step precedes said selection step.

5. The method for converting performance metric data of claim 2 wherein the plurality of attributes include NODE.

6. The method for converting performance metric data of claim 2 wherein the plurality of attributes include APP.

7. The method for converting performance metric data of claim 2 wherein sorting the first dataset includes selecting records from the first data set and grouping the selected records into discrete tables wherein the selected records each have a first one of the plurality of attributes and wherein a value of the first one of the plurality of attributes is the same for each selected record.
8. The method for converting performance metric data of claim 7 wherein the first one of the plurality of attributes is APP.
9. The method for converting performance metric data of claim 3 wherein the plurality of retrieved UDF files are produced by one or more of the plurality of nodes.
10. The method for converting performance metric data of claim 3 wherein at least one of the plurality of retrieved UDF files includes performance metric data from a first time segment and another one of the plurality of retrieved UDF files includes performance metric data from a second time segment.
11. The method for converting performance metric data of claim 10 wherein the first time segment is different from the second time segment.
12. The method for converting performance metric data of claim 10 wherein building the plurality of performance data tables includes:
 - (a) selecting at least one of the discrete tables and reading the records of the at least one discrete table,
 - (b) constructing output records by (i) combining a selected first group of the plurality of attributes with a selected second group of the plurality of attributes, the second group being selected according to the value of the value of the first one of the plurality of attributes, and (ii) for each record of the at least one data table, mapping a value of a second one of the plurality of attributes to one of the second group of attributes according to a value of a first one of the first group of attributes, and

(c) summarizing the output records according to a third group of the plurality of attributes.

13. The method for converting performance metric data of claim 12 wherein the first group of the plurality of attributes includes APP, DATE, DATETIME, HOUR, INSTANCE, NODE, PARM QUARTER, and TIME.

14. The method for converting performance metric data of claim 13 wherein the first one of the first group of attributes is PARM.

15. The method for converting performance metric data of claim 14 wherein the second one of the plurality of attributes is METRIC.

16. An apparatus for converting performance metric data from a plurality of nodes, produced by a plurality of types of collection agents resident on said plurality of nodes, from UDF files to a second format, said apparatus comprising:

a data processor programmed to:

retrieve a plurality of UDF files from a corresponding plurality of collection agents, each UDF file including a plurality of records comprising performance metric data corresponding to one of the plurality of nodes, the performance metric data being separated into a plurality of fields;

reformat each UDF file and map each UDF file to a first data set;

sort the first dataset including group the records by attribute; and build a plurality of performance data tables using the sorted first dataset.

17. The apparatus of claim 16 wherein said data processor is programmed to reformat each UDF file by parsing the data fields of each record and mapping the parsed data fields to a plurality of attributes.

18. The apparatus of claim 17 wherein said data processor is programmed to sort the first dataset by selecting records from the first data set and grouping the selected records into discrete tables wherein the selected records each have a first one of the plurality of attributes and wherein a value of the first one of the plurality of attributes is the same for each selected record.

19. The apparatus of claim 18 wherein said data processor is programmed to build the plurality of performance data tables by:

(a) selecting at least one of the discrete tables and reading the records of the at least one discrete table,

(b) constructing output records by (i) combining a selected first group of the plurality of attributes with a selected second group of the plurality of attributes, the second group being selected according to the value of the value of the first one of the plurality of attributes, and (ii) for each record of the at least one data table, mapping a value of a second one of the plurality of attributes to one of the second group of attributes according to a value of a first one of the first group of attributes, and

(c) summarizing the output records according to a third group of the plurality of attributes.

20. The apparatus of claim 17 wherein the plurality of retrieved UDF files are produced by one of the plurality of types of collection agents.

21. The apparatus of claim 20 wherein the plurality of retrieved UDF files include performance metric information for a selected date.

22. The apparatus of claim 20 wherein the plurality of retrieved UDF files are produced by one or more of the plurality of nodes.

23. The apparatus of claim 20 wherein at least one of the plurality of retrieved UDF files includes performance metric data from a first time segment and another one of the plurality of retrieved UDF files includes performance metric data from a second time segment.

24. The apparatus of claim 23 wherein the first time segment is different from the second time segment.

25. A computer program product comprising a computer useable medium having program logic recorded thereon for use with a data processor to convert performance metric data, produced by a plurality of collection agents, from UDF files to a second format, the plurality of collection agents being resident on a plurality of nodes, said computer program logic comprising:

computer readable means for retrieving a plurality of UDF files from a corresponding plurality of collection agents, each UDF file including a plurality of records containing performance metric data corresponding to one of the plurality of nodes, the performance metric data being separated into a plurality of fields;

computer readable means for reformatting each UDF file and mapping each UDF file to a first data set;

computer readable means for sorting the first dataset including grouping the records by attribute; and

computer readable means for building a plurality of performance data tables using the sorted first dataset.

26. The computer program product of claim 25 wherein said computer readable reformatting means includes means for parsing the data fields of each record and for mapping the parsed data fields to a plurality of attributes.

27. The computer program product of claim 26 wherein said computer readable means for sorting includes computer readable means for selecting records from the first data set and grouping the selected records into discrete tables wherein the selected records each have a first one of the plurality of attributes and wherein a value of the first one of the plurality of attributes is the same for each selected record.

28. The computer program product of claim 27 wherein said computer readable means for building includes:

(a) computer readable means for selecting at least one of the discrete tables and reading the records of the at least one discrete table,

(b) computer readable means for constructing output records by (i) combining a selected first group of the plurality of attributes with a selected second group of the plurality of attributes, the second group being selected according to the value of the value of the first one of the plurality of attributes, and (ii) for each record of the at least one data table, mapping a value of a second one of the plurality of attributes to one of the second group of attributes according to a value of a first one of the first group of attributes, and

(c) summarizing the output records according to a third group of the plurality of attributes.

29. A system for processing and analyzing performance metric data for input to a data analysis/reporting tool, said system comprising:

a plurality of nodes, each node having a collection agent that collects performance metric data and compiles the performance metric data into UDF files, each UDF file including a plurality of records containing performance metric data corresponding to one of said plurality of nodes, the performance metric data being separated into a plurality of fields; and

a data analysis computer including a data processor programmed to:

retrieve a plurality of UDF files from a corresponding plurality of collection agents, each UDF file including a plurality of records containing performance metric data corresponding to one of the plurality of nodes, the performance metric data being separated into a plurality of fields;

select the UDF files produced by one of the plurality of collection agents;

reformat each UDF file and map each UDF file to a first data set;

sort the first dataset including group the records by attribute;

and build a plurality of performance data tables using the sorted first dataset.

30. The system of claim 29 wherein the plurality of nodes include a plurality of computers.

31. The apparatus of claim 30 wherein said data processor is programmed to reformat each UDF file by parsing the data fields of each record and mapping the parsed data fields to a plurality of attributes.

32. The apparatus of claim 31 wherein said data processor is programmed to sort the first dataset by selecting records from the first data set and grouping the selected records into discrete tables wherein the selected records each have a first one of the plurality of attributes and wherein a value of the first one of the plurality of attributes is the same for each selected record.

33. The apparatus of claim 32 wherein said data processor is programmed to build the plurality of performance data tables by:

- (a) selecting at least one of the discrete tables and reading the records of the at least one discrete table,

- (b) constructing output records by (i) combining a selected first group of the plurality of attributes with a selected second group of the plurality of attributes, the second group being selected according to the value of the value of the first one of the plurality of attributes, and (ii) for each record of the at least one data table, mapping a value of a second one of the plurality of attributes to one of the second group of attributes according to a value of a first one of the first group of attributes, and

- (c) summarizing the output records according to a third group of the plurality of attributes.

34. The apparatus of claim 31 wherein the plurality of retrieved UDF files are produced by one of the plurality of types of collection agents.

35. The apparatus of claim 34 wherein the plurality of retrieved UDF files include performance metric information for a selected date.

36. The apparatus of claim 34 wherein the plurality of retrieved UDF files are produced by one or more of said plurality of nodes.

37. The apparatus of claim 34 wherein at least one of the plurality of retrieved UDF files includes performance metric data from a first time segment and another one of the plurality of retrieved UDF files includes performance metric data from a second time segment.

38. The apparatus of claim 37 wherein the first time segment is different from the second time segment.