Sizing for SAP Data Mining

Version 1.3

June 2004

Copyright 2004

TABLE OF CONTENTS

About this document

This document is developing sizing guidelines for SAP data mining engines. It is based on results of an ongoing project for systematically measuring the resource consumption (i.e. disk space, CPU and memory consumption) of

a) Decision tree for data classification (C4.5 algorithm)

b) Clustering for data segmentation (k-Means Algorithm)

c) Weighted score tables for scoring of records

d) ABC Analysis for classification

e) Association Analysis for market basket analysis (A-Priori Algorithm)

As the precise resource consumption of each method is completely dependent on many parameters such as the size and content of the data to be processed, and mining model parameters, the document conceptually describes the major resource consuming building blocks of each algorithm. This allows estimating the resource consumption for each mining method for a given customer setting, and to derive according hardware sizing requirements. For details of the concepts of each algorithm, please refer to the according documentation.

As the most resource consuming process in data mining is the training of data mining models, this document focuses on resource consumption of training, and does not cover prediction and evaluation.

The sizing measurements exclusively address the pure data mining functionality, and do not include the standard BW data (pre-) processing, such as extraction of data from the BW query.

Target Audience

This document is designed for readers who have the basic understanding of Data Mining. Reader is expected to be aware of Data Mining Tools. Elementary knowledge of Data Mining algorithms would help in understanding the document's content better.

1 Sizing for Decision Tree

A decision tree is used as a classifier for determining an appropriate action or decision (among a predetermined set of actions) for a given case. A decision tree helps one to effectively identify the factors one must consid er and how each factor has historically been associated with different outcomes of the decision.

Decision trees have become one of the most popular data mining tools. Their visual presentation makes the decision trees very easy to read, understand and ass imilate information from it. They are called decision trees because the resulting model is presented in the form of a tree structure. Decision trees are most commonly used for classification, that is, predicting to which group a particular case belongs.

1.1 Disk Space Consumption

Each data mining engine writes temporary data and result data to database tables. Consequently, the database should be large enough to keep such data. The real table space requirements for data mining engines depend on many parameters , and therefore it is impossible to give a precise quantification of the disk space consumption. However, to estimate the extra table space needed by a decision tree engine, we describe how to estimate the space needed by the different extra tables created by a decision tree engine with potentially high data volume:

a) Extract Table ET. Before BW release 3.5, t he input data for training, evaluation, and prediction for all data mining engines is delivered by the execution of BW Queries (defined in BeX Analyzer). Before a training run, an evaluation run, or prediction run is started, the query result is written to an Extract Table (ET) in the database, and the data mining engines take their input data from this extract table ET. The size of ET completely depends on the number of attributes and the number of records returned by the query. To compute the size of extract table ET , the number of records has to be multiplied by the size of a single record. The size of a single record has to be computed as the sum of the sizes for each characteristic and key figure contained in the record. Please r efer Appendix A for more information on estimating the size of the extract table.

   However, from BW release 3.5, the input data comes from APD and the size of the extract table depends on the model fields. The sum of sizes of each field (refer to the corresponding InfoObject of the model fields) should give the approximate width of a single row of this extract table. Unlike previous releases, this extract table is deleted immedia tely after training.

b) Training Table T1. To optimize training of a data mining model, it does not make sense to do the training on the original data set, but to transform the original data to so -called enumerated data of highly reduced size. For example , values such as “male” and “female” could simply be encoded as enumerated data 0 and 1, making the internal processing much more efficient.
   The Training Table T1 keeps the enumerated data in the database. The size of training table T1 is estimated as follows:

   D -> number of discrete fields

   C -> number of continuous fields

   Size of record in T1: L = 26 Bytes + D * 5 Bytes + C * 12 Bytes

   Disk Space used by table T1

   SIZE(T1) = L * number of records in ET

c) Training Table T2. A second enumeration table T2 is used only if sampling option is used . In this case, T2 keeps the samples taken from T1. Consequently, the size of T2 depends on the windowing parameters. The maximum number of records would be:

MAX_SIZE(T2) = L Bytes * Max Sample Size (%) /100 * *number of records in T1*

where "Max Sample Size (%)" denotes the model parameter defined in the sampling option .

d) Decision Tree Result Tables. The decision tree engine uses a set of database tables to keep the decision tree results. The following tables are affected:

i. **Tree Table** to keep the decision tree nodes. The space needed in this table for a decision tree is

278 Bytes * number of decision tree nodes in the trained model.

The number of decision tree nodes created is difficult to estimate, as it depends on the shape of the tree which depends on the data and the model parameters. For a perfectly balanced tree of depth *n* (assuming that a single node has depth n=1) and a branching value of *m* for each inner node, the number of nodes can be estimated by the formula $(m^n - 1) / (m-1)$. For example, a binary split tree up to 10 levels (*m* = 2 and *n* = 10), there will be 1023 records in this table out of which 512 are leaf nodes. Note that this formula can only give estimation, as it assumes that there are equal splits at every level and that all leaf nodes have the same depth in the tree, which is unrealistic for typical models.

ii. **Leaf Table** to keep the leaf nodes of the decision tree. The space needed in this table for a decision tree is

(*120 Bytes * NumberOfLeafTreeNodes * NumberOfValidClassValues*), where

NumberOfLeafTreeNodes denotes the number of decision tree leaf nodes in the trained model, and *NumberOfValidClassValues* denotes the number of valid values for the predictable model field. For the very large decision tree discussed above with depth 8 and a branching value of 5, the number of leaf nodes would be 390,625 and thus the space needed would be 89 MB, assuming that there are 2 different valid values for the predictable model field. Again, this table space consumption is rather a "worst case".

iii. **Rules Header Table** to keep rules header data. The space needed in this table for a decision tree is (*115* Bytes * *NumberOfLeafTreeNodes*).

iv. **Rules Detail table** to keep the rules contained in the decision tree. The size of each record to keep a rule is 195 Bytes. The number of records in the table will be the number of nodes in the unique paths leading from the root to all leaf nodes. Consequently the size of the rules detail table is given by:

$$195 Bytes * \sum_{i=1}^{depthofthetree} \left(i * numberOfLeafNodes(i) \right).$$

where *numberOfLeafNodes(i)* is the number of leaf nodes that can be found at level i of the tree (level i=1 being the root node). Hence if more leaf nodes are formed deeper in the tree, this will result in more number of records. Note that the number of records in this table is higher than the number of nodes in the tree, as nodes at upper levels are "shared" by many paths.

Lifetime of tables

The table space consumption of a data mining method varies over time, as it depends on the phase of the training.

The extract table ET is deleted automatically when the data source for training is deleted in the model (to be done manually in the data mining workbench, or automatically on deletion of the model).

Tables T1 and T2 are temporary tables and are deleted automatically after training.

Decision Tree Result Tables are deleted when the model is reset in the data mining workbench.

1.2 Memory Consumption

During various stages, there are memory-consuming operations that are either temporary or permanent.

a) During enumeration, the entire enumerated data is held in memory in an enumeration table for sorting. Hence the maximum memory size will be equal to the size of the enumerated table (training table T1 described in the disk space section). After writing into the enumerated table, this memory is freed.

   Example: Assume a training table T1 filled by a query delivering 10 characteristics and 10 key figures, to be used to train a decision tree. Assume the query delivers 1 million records (with 20 attributes each). According to the disk space consumption of T1, the size of T1 is given by

   Size (T1) = 1 Million * (26 Bytes + 10 * 5 Bytes + 10 * 12 Bytes) = 190 Megabytes

b) During tree formation phase, the main memory consumers are the tree, tree node distribution, rules header and detail tables. Memory estimation is very difficult for these cases as they are entirely dependent on data distributions and the model parameters (especially the stopping conditions like minimum leaf cases and accuracy). The more branches and nodes a tree has, the more memory is consumed. The following rules of thumb apply:

   i. If the minimum leaf cases are set low and minimum accuracy is set high, then the depth of the tree increases.

   ii. If the cardinality of discrete fields is high, then there are many branches in the tree, leading to increased width of the tree. However, a model consisting of high cardinality discrete fields tend to produce shallow trees.

   iii. A model consisting of continuous fields with random distribution produces deep trees.

c) For each continuous field, all possible distinct values and their frequency in the data during the calculation is kept in memory, to determine the decision node for a split. This is done at every step of decision-making and the maximum memory consumed will be in the first step. The memory consumption of this step normally is typically much lower than that consumed by the enumeration table described in section a) before.

1.3 CPU Consumption

This section qualitatively describes the impact of model and data characteristics on CPU consumption. The major CPU consumption takes place in the following steps:

a) Determination of frequency distribution for all fields. The more fields a model has, the higher the CPU consumption is.

b) Determination of Decision Node. To determine the next decision node a lot of arithmetic operations have to be performed. The more fields are used, and the higher the cardinality of discrete fields, the more complex the processing is. In addition, the processing time for the split of continuous fields is higher than the processing time for discrete fields.

   Steps a) and b) are recursively processed, and hence if the tree grows deeper, CPU consumption increases and so does the runtime.

c) Pruning of the tree. Pruning of the tree needs to more complex processing. On one hand it leads to a lot of arithmetic operations, and thus leads to major CPU consumption, as pruning evaluates each branch of the tree. On the other hand, each evaluation needs to access the value distribution frequency of the predictable field to be read from the database. Consequently, pruning also leads to

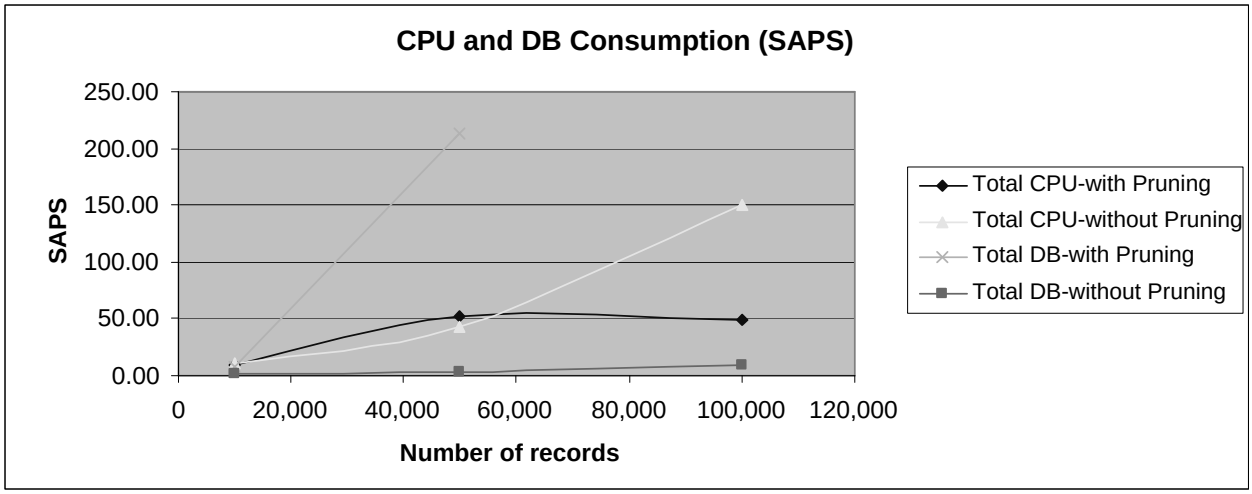
significant database access.

1.4 Example Sizing Figures

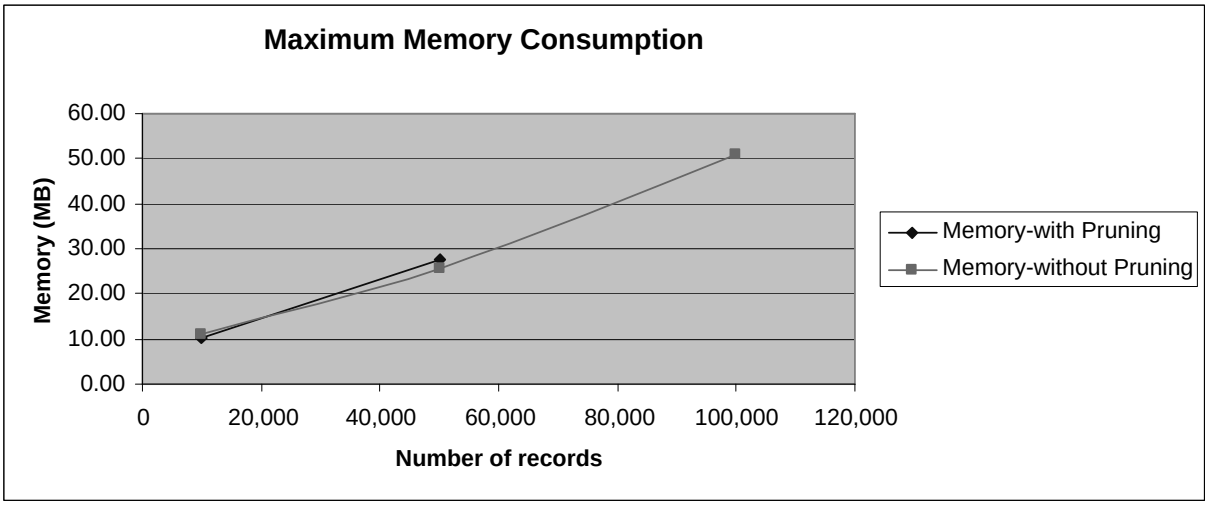
The following table shows example figures of the memory and CPU consumption for different data sizes and model parameters, and shows the CPU time and database access time.

Number of records	Minimum leaf cases	Perform pruning	Total CPU[SAPS]*	Total DB [SAPS]	Maximum memory consumption [MB]
10,000	100	No	10.52	1.25	11.22
50,000	100	No	43.06	3.39	25.49
100,000	100	No	149.91	9.95	50.98
10,000	100	Yes	9.73	7.79	10.29
50,000	100	Yes	52.45	212.46	27.53
100,000	1000	Yes	48.80	72.86	18.35

* Refer Glossary



The values show that the decision tree algorithm is CPU bound in most cases due to the tree processing. However, using pruning of the tree leads to many database accesses, and in case of less complex trees this may even lead to database access bound overall runtime.



Regarding memory consumption, the figures show that the more records are used for training, the higher the memory consumption is, due to the increased size of training table T1. In addition, the higher the minimum leaf cases and if no pruning is used, the more memory is consumed for the storage of the nodes. If pruning is used, the algorithm can merge nodes leading to a lower number of nodes and thus reduced memory consumption, at the cost of increased CPU consumption.

2 Sizing Estimate for Clustering

Decision trees have become one of the most popular data mining tools. Their visual presentation makes the decision trees very easy to r ead, understand and assimilate information from it. They are called decision trees because the resulting model is presented in the form of a tree structure. Decision trees are most commonly used for classification, that is, predicting to which group a part icular case belongs.

2.1 Disk Space Consumption

To estimate the extra table space needed by clustering, we describe how to estimate the space needed by the different extra tables created by clustering with potentially high data volume:

a) Extract Table ET. See section 1.1 a)

b) Training Table T1. To optimize training of a data mining model, it does not make sense to do the training on the original data set, but to transform the original data to so -called enumerated data of highly reduced size. For example, values such as “male” and “female” could simply be encoded as enumerated data 0 and 1, making the internal processing much more efficient.
The Training Table T1 keeps the enumerated data in the database. Table T1 is deleted automatically after training.

   The size of training table T1 is estimated as follows:

   D -> number of discrete fields

   C -> number of continuous fields

   Size of record in T1: L = 10 Bytes + D * 5 Bytes + C * 12 Bytes

   Disk Space used by table T1: SIZE(T1) = L Bytes * number of records in ET

c) Key Table KT. The table KT keeps the cluster assignments for all input records after the training. It contains the key fields of the extract table ET and some clustering result fields. The number of records in KT is equal to the number of records in T1. Each record in KT has size

   L_TK = 26 Bytes + KeyLength(ET)

   where KeyLength(ET) is the sum of all key field sizes in ET.

   The disk space of key table KT is

   Size(KT) = L_TK Bytes * number of records in KT

d) Clustering Result Tables. The following result tables are used to store the clustering results after training. The data volume needed to store these results highly depends on the data distribution and may occupy more space if field cardinality is very high.

   i. Cluster Distribution Table CDT. The cluster definition table CDT contains distribution info of each value of each field in each cluster. The size of CDT is given by

      Size(CDT) = 201 Bytes * (number of clusters) * (number of distinct field values of all discrete fields and ordered fields + number of binning intervals for each continuous field).

      Example: Assume a model is defined with two discrete fields *Gender* and *Marital Status*, and a single continuous field *Income*. Gender has two values (Male & Female), Marital Status has four values (Married, Divorced, Single, Widowed). The number of binning intervals for Income is 20. The number of clusters is 10. Then Size(CDT) is given by

      Size(CDT) = 201 Bytes * 10 * (2 + 4 + 20) = 0.05226 MB

   ii. Cluster Mean Detail Table CMDT. The cluster mean detail table contains cluster parameters. The size of CMDT is given by

      size(CMDT) = 96 Bytes * (number of clusters) * (number of distinct discrete field values of all discrete or discretised fields + number of continuous fields)

Lifetime of tables

The table space consumption of a data mining method varies over time, as it depends on the phase of the training.

The extract table ET is deleted automatically when the data source for training is deleted in the model(to be done manually in the data mining workbench, or automatically on deletion of the model).

Tables T1 is a temporary table and are deleted automatically after training.

Clustering Tables are deleted when the model is reset in the data mining workbench.

2.2 Memory Consumption

In Clustering, the entire input data is not loaded completely into memory but is read piecewise from the database table. In the core logic, there are multiple database accesses done, logically retrieving one record at a time and processing the same.

During the training, all result tables are kept in memory. The major contributor among these result tables are the Mean Detail table CMDT, the Cluster Distribution table CDT, and the Key table KT. The peak memory consumption occurs in the last stages of training when all these tables are completely filled.

Note that the memory consumption and the disk space consumption of the tables are different, as the fields kept in memory are typically a subset of the database table fields.

The memory used by Cluster Mean Detail Table can be calculated as follows:

MemorySize(CMDT) = 69 Bytes * (number of clusters) * (number of distinct field values of all discrete or discretised fields + number of continuous fields)

The memory used by Cluster Distribution table CDT can be calculated as follows:

MemorySize(CDT) = 141 Bytes * (number of clusters) * (number of distinct field values of all discrete fields and ordered fields + number of binning intervals for each continuous field)

The memory used Key table can be calculated as follows:

MemorySize(KT) = 126 Bytes * number of records in the input data (ET)

The memory sizes of CMDT and CDT purely depend on the number of clusters and also on the cardinality of the fields and not on the data sizes , but typically leads to memory consumption of few hundre d kilobytes.

The memory consumption of the key table in creases linearly with the records to be processed and in most cases dominates the overall memory consumption of the cluster result tables.

Example: Assume a model with two discrete field values of thr ee distinct values each, and eight continuous fields with 10 binning intervals each. The model has 10 clusters and was trained with 1 million records. Then the total memory consumption of the result tables discussed above is around 120 MB.

2.3 CPU Consumption

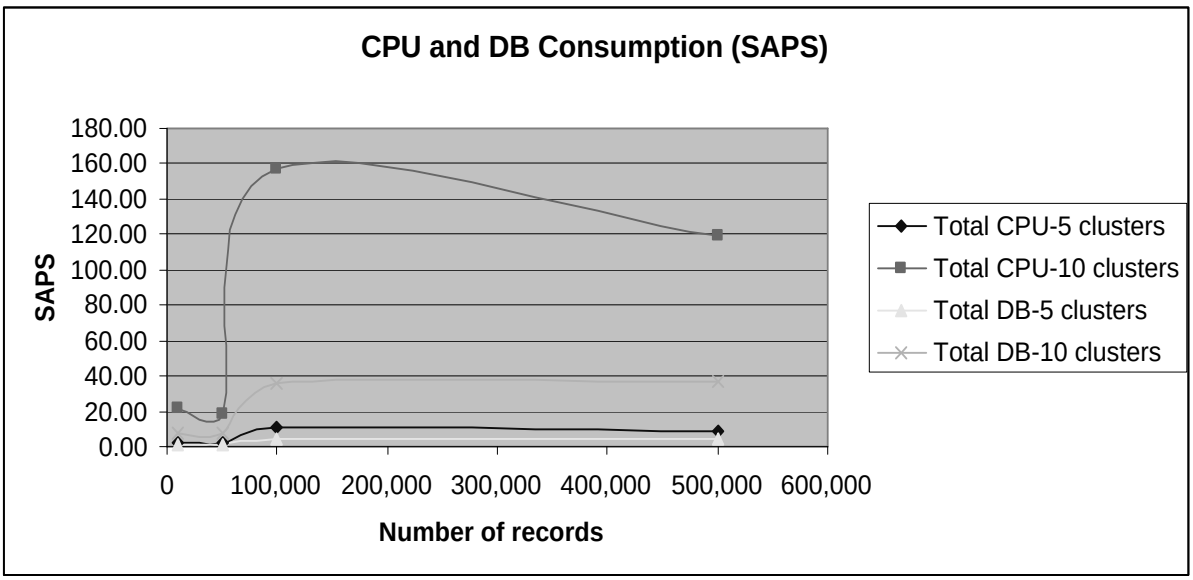
The clustering algorithm implemented does not contain very complex arithmetic operations. On the other hand, access to the database tables is limited to simple scans of training table T1 and does not contain I/O intensive random access queries, and results are written to the in -memory tables. In typical settings , clustering is typically CPU bound.

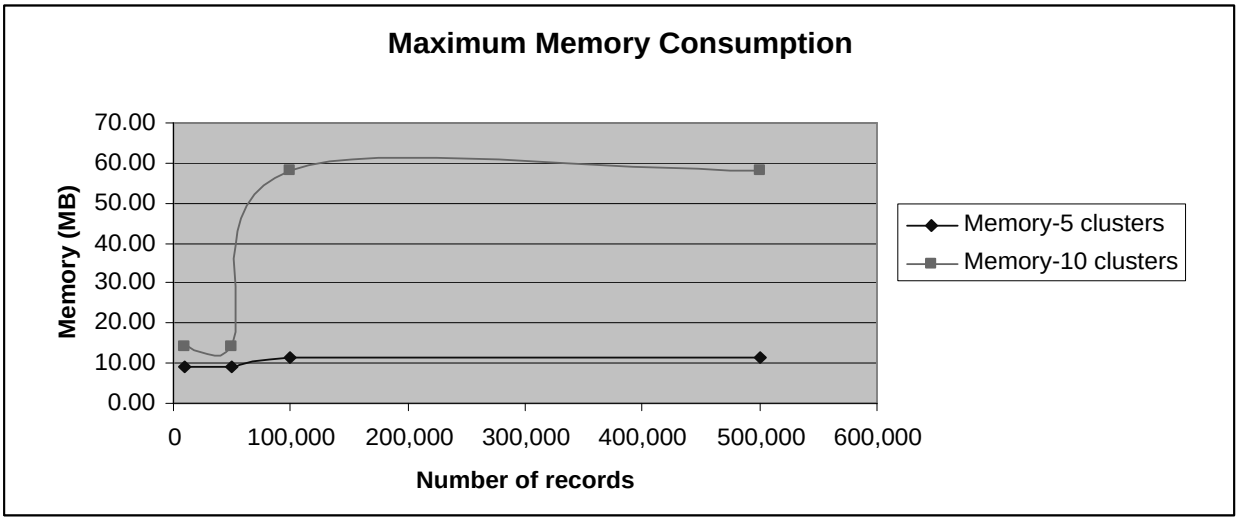
2.4 Example Sizing Figures

In this section, we have given example s of sizing figures we obtained from experimental results. Eight different clustering models were cre ated with varying number of input records , and two different maximum number of clusters (5 and 10), to illustrate the CPU and memory consumption behavior.

The figures given below illustrate CPU consumption and memory consumption of the clustering method under varying number of input records and a maximum number of clusters of 5 and 10, respectively.

Number of records	Number of clusters	Total CPU [SAPS]	Total DB [SAPS]	Maximum memory consumption [MB]
10,000	5	1.88	0.99	9.18
50,000	5	2.25	0.91	9.18
100,000	5	10.91	3.85	11.22
500,000	5	9.07	3.83	11.22
10,000	10	21.81	7.34	14.27
50,000	10	18.10	7.31	14.27
100,000	10	157.32	36.20	58.11
500,000	10	118.73	36.37	58.11





The results show the following

i. There is a steady increase in the CPU and memory consumption as the number of input records increases. Major contributor to the memory consumption is the size of the cluster assignment table, which is also the reason why the number of clusters has minor impact on the overall memory consumption.

ii. The CPU consumption increases with increasing number of clusters, due to the increasing processing complexity.

3 Sizing for Weighted Table Scoring

Weighted table scoring algorithms are simple record transformation and do not involve complex calculations. All calculation information is specified in the model definition itself – data fields, values to be considered for each data field and binning intervals in case of continuous fields. Based on this input, the algorithm calculates the score for each input record. Hence normally, we expect linear behavior with data sizes in terms of Disk space, CPU and Memory as explained in the following sections.

3.1 Disk Space Consumption

Extract Table ET. See section 1.1 a)

a) Score Assignment Table KT. The table KT keeps the score assignments for all input records after the training. It contains the model key fields from the extract table ET and corresponding score value. The number of records in KT is equal to the number of records in ET. Each record in KT has size

   L_KT = 12 Bytes + Length of all infoobjects corresponding to model key fields.

   The disk Space of Score Assignment KT is

   Size (KT) = L_KT Bytes * number of records in KT

   For example, if 0CUSTOMER was used as the key, the length of the Scoring assignment table is (12 Bytes + 10 bytes = 22 Bytes) and if the extract table had 1 million customers, then disk space consumption would be about 22 MB.

b) Scoring result tables . The following result tables are used to store the scoring results after training. The space consumption for these tables is not very large and does basically depend on the cardinality of discrete fields and on the number of binning intervals for continuous fields. The overall disk space consumption is given by

   (574 Bytes + 720 Bytes * number of trained/predicted sources assigned to the model) *

   (number of defined values for all discrete fields + number of defined intervals for all continuous fields)

Lifetime of tables

The extract table ET is deleted automatically when the data source for training is deleted in the model (to be done manually in the data mining workbench, or automatically on deletion of the model).

Scoring Tables are deleted when the model is reset in the data mining workbench.

3.2 Memory Consumption

The memory consumption is small, as basically the model parameters and result statistics are kept in memory for a quick computation of the predicted vales, and this memory consumption is dependent completely on the model fields and parameters defined, and their data distribution like number of values per field.

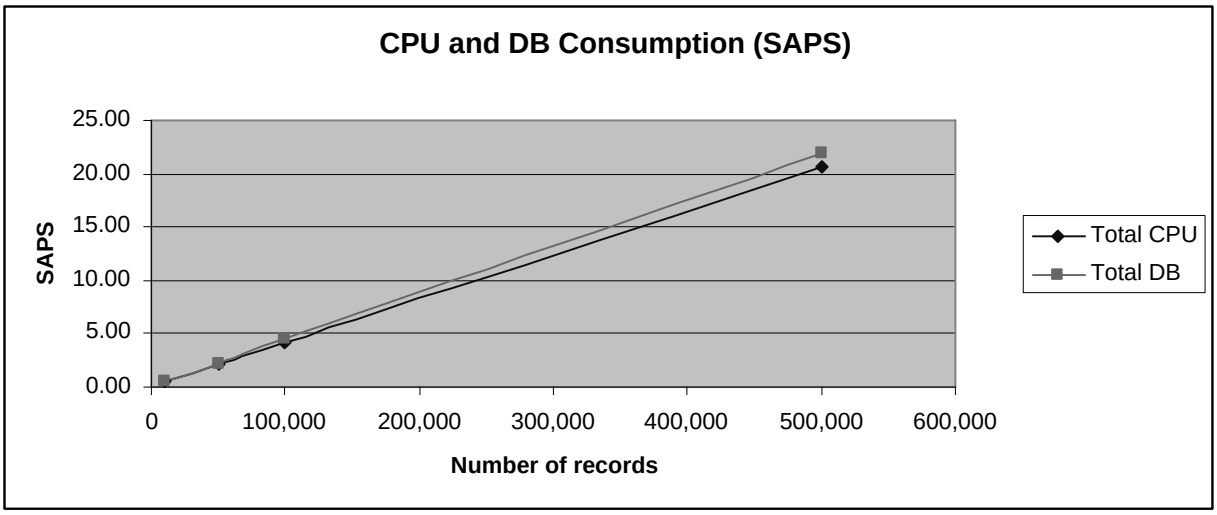
3.3 CPU Consumption

The CPU consumption of weighted table scoring is rather low, as weighted table scoring requires simple arithmetic operations and in-memory table operations. However, as database access is limited to a simple table scan and plain inserts into the assignment and result tables , the runtime of weighted table scoring typically is CPU bound.

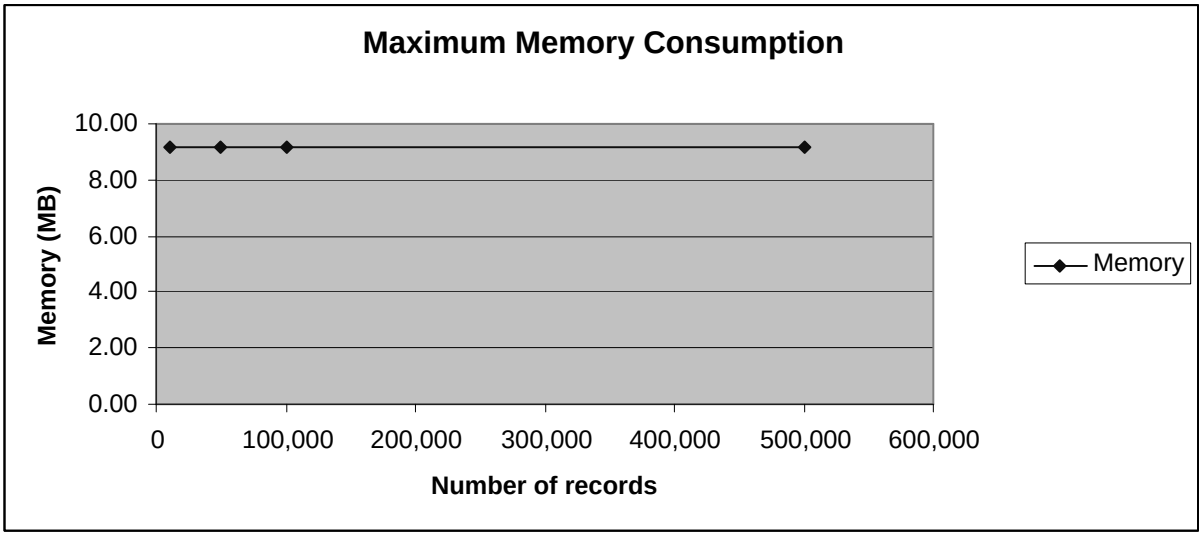
3.4 Example Sizing Figures

The following table shows example figures of the memory and CPU consumption for different data sizes and shows the CPU time and database access time on the overall runtime.

Number of records	Total CPU [SAPS]	Total DB [SAPS]	Maximum memory consumption [MB]
10,000	0.55	0.51	9.18
50,000	2.17	2.26	9.18
100,000	4.19	4.44	9.18
500,000	20.59	21.89	9.18



The figures given below illustrate the impact of CPU consumption and database access on the overall runtime. The figures show that in this example the weighted table scoring method is equally CPU bound and DB bound.



As expected, the memory consumption was constant for all these tests and was around 10 MB because basically the model parameters and result statistics are kept in memory for a quick computation of the predicted vales, and this memory consumption is dependent completely on the model fields and parameters defined, and their data distribution like number of values per field.

4 Sizing for ABC Analysis

The *ABC Classification* is a frequently used analytical method to classify objects (Customers, Products or Employees) based on a particular measure (Revenue or Profit). For example, one can classify one's customers into three classes A, B and C according to the sales revenue they generate.

ABC classification allows one to classify one's data based on specified classification rules. The data to be classified is generated by a query in the SAP BW. The classification rules refer to a single key figure value in one's data and implicitly specify which absolute or relative key figure values map to which classes.

4.1 Disk Space Consumption

Major disk space is consumed by the following database tables:

a) Extract Table ET. See section 1.1 a)

b) ABC Assignment Table AAT. The table AAT keeps the assignments of A,B,C classes for all input records after the training. As a rough estimate, the size of AAT is given by

   Size(AAT) = Number of records to be processed * (71 Bytes + size(ABC_Class) + size(ClassificationCriterion) + size(ClassifiedObject))

   where size(ABC_Class) is the size of the info object specified as ABC class, size(ClassificationCriterion) is the size of the info object specified as classification criterion, and size(ClassifiedObject) is the size of the info object specified as classified object in the model.

Lifetime of tables

The extract table ET is deleted automatically when the data source for training is deleted in the model (to be done manually in the data mining workbench, or automatically on deletion of the model).

4.2 Memory Consumption

The memory consumption of ABC analysis is not very high (up to few megabytes), as only the model metadata and basic statistics are loaded into memory. Experimental results have shown that the training of an ABC model with 10'000 up to 50'000 records led to a constant memory consumption of around 8 MB.

4.3 CPU Consumption

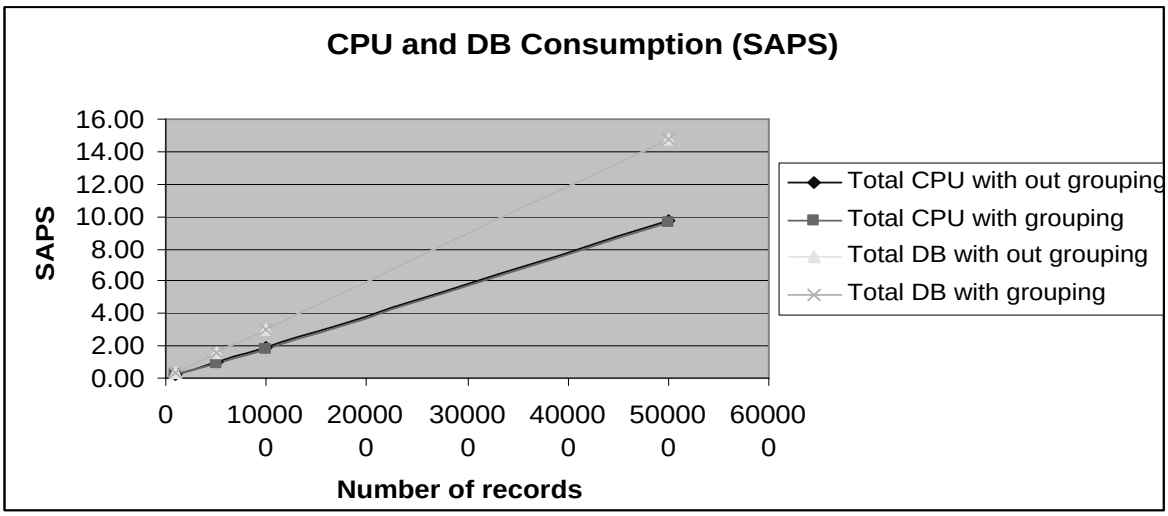
As for weighted table scoring, the CPU consumption of ABC analysis is rather low, as it requires simple arithmetic operations and in-memory table operations. First experimental results have shown that the impact of CPU consumption and database access on the overall runtime is quite balanced (50%/50%), which is due to the less processing complexity as compared to weighted table scoring.

4.4 Example Sizing Figures

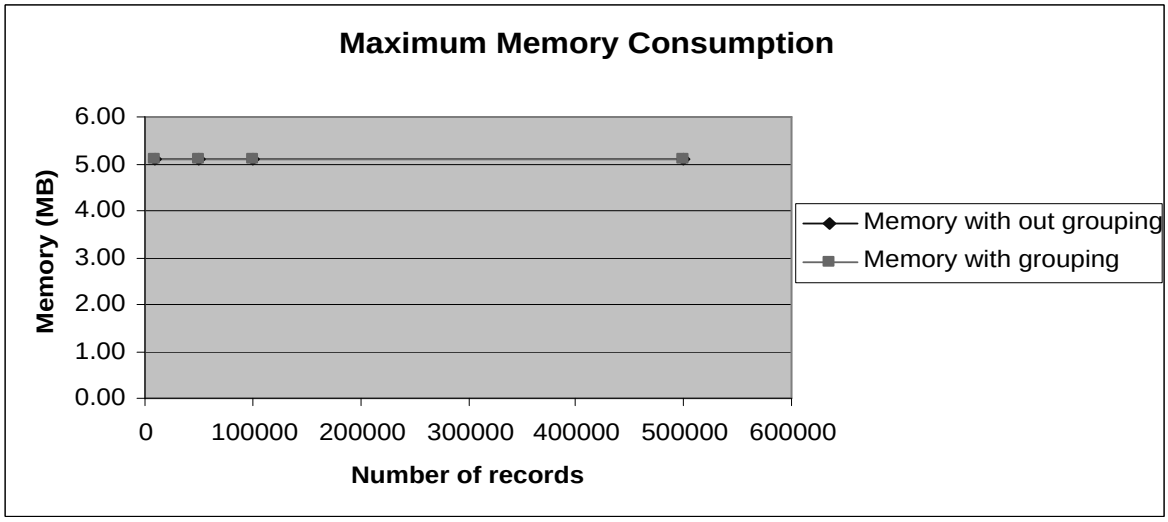
The following table shows example figures of the memory and CPU consumption for different data sizes with and with out grouping and shows the CPU time and database access time on the overall runtime.

Number of records	Grouping	Total CPU [SAPS]	Total DB [SAPS]	Maximum memory consumption [MB]
10000	No	0.23	0.38	5.10
50000	No	1.00	1.51	5.10
100000	No	1.95	3.01	5.10
500000	No	9.70	14.72	5.10
10000	Yes	0.22	0.37	5.10
50000	Yes	0.93	1.55	5.10
100000	Yes	1.81	3.03	5.10
500000	Yes	9.59	14.78	5.10

The figures given below illustrate the impact of CPU consumption and database access on the overall runtime.



Regarding memory consumption, the figures show that the more memory is required when gro uping is used.



5 Sizing for Association Analysis

Association Analysis uncovers the hidden patterns, corre lations or casual structures among a set of items or objects. For example, Association Analysis enables one to understand what products and services customers tend to purchase at the same time. By analyzing the purchasing trends of one's customers with *Association* Analysis, one can predict their future behavior. It is also commonly referred to as "association discovery".

The association analysis typically is the most resource consuming data mining method, as it inherently gets millions of transactional data that need to be evaluated in multiple passes, and a lot of intermediate results have to be stored in each pass.

This section analyses the resource consumption of association analysis. As for all other mining methods, an association algorithm starts as soon as the data is extracted from the BW Query.

5.1 Disk Space Consumption

Estimating the disk space consumption of association analysis is very complex, as it largely depends on the data (the rules in the data), and the model parameters like minimum support, confidence, and lift. The following tables are the major space consumers:

a) Extract Table ET. See section 1.1 a)

b) Enumeration table. The extracted data is transformed into the enumerated table by mapping the transaction and item id to compact data types. This enumeration reduces the width of the input data so that lesser space is consumed and internal processing is less complex. Also, redundant or invalid data is also removed during enumeration. The size of the enumeration table is given by (32 Bytes * Number of records in ET).

c) Result tables. The association discovery internally has to count occurrences of item-sets of different sizes in the enumeration table. Therefore, a lot of intermediate and permanent result tables have to be created. The size of these tables highly depends on the number of input records, number of unique items, the minimum support and confidence specified in the model, and the rules identified.

Lifetime of tables

The extract table ET is deleted automatically when the data source for training is deleted in the model (to be done manually in the data mining workbench, or automatically on deletion of the model).

Enumerated table is temporary table and is deleted automatically after training.

Association result tables are deleted when the model is reset in the data mining workbench.

5.2 Memory Consumption

The association analysis uses dynamically sized buffers for the processing of the data. For millions or even tens of millions records to be processed these buffers may need several hundred megabytes of memory for an efficient execution and acceptable runtimes.

5.3 CPU Consumption

Association analysis is highly demanding on both the database and the CPU, and therefore the impact of CPU processing and database access on the overall runtime is quite balanced.

5.4 Example Sizing Figures

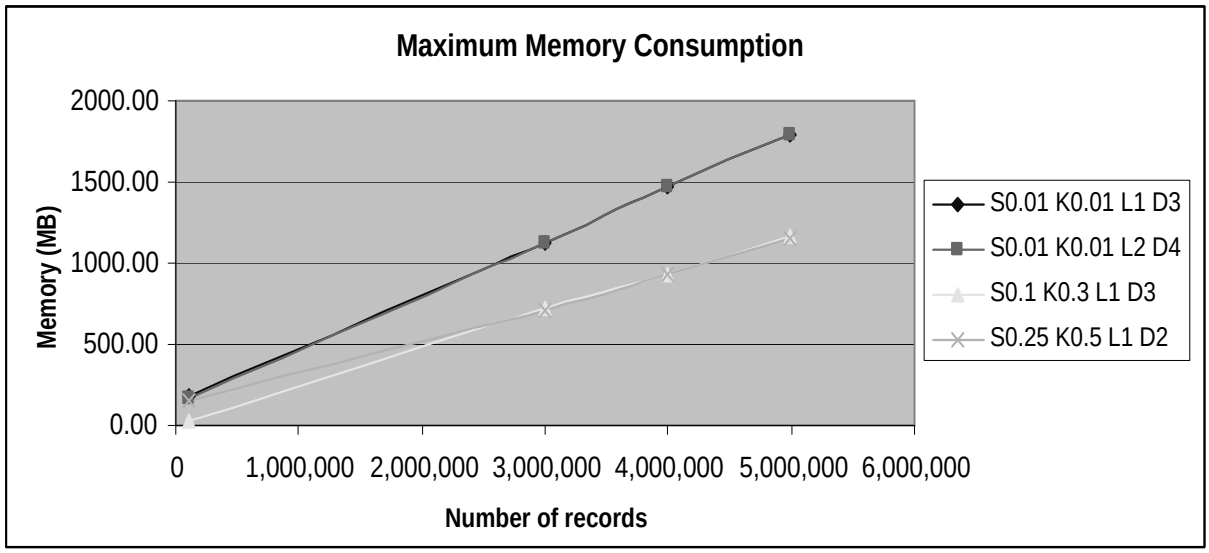
The following figures illustrate the disk space and memory consumption for different association analysis models covering a spectrum of model parameters.

Number of records	Support	Confidence	Leading items	Dependent items	Total CPU [SAPS]	Total DB [SAPS]	Maximum memory consumption [MB]
99,979	0.01	0.01	1	3	5.18	6.94	176.68
2,999,746	0.01	0.01	1	3	42.63	33.22	1121.17
3,993,689	0.01	0.01	1	3	56.73	43.54	1478.96
4,993,431	0.01	0.01	1	3	102.98	52.92	1797.25
99,979	0.01	0.01	2	4	5.12	5.68	170.34

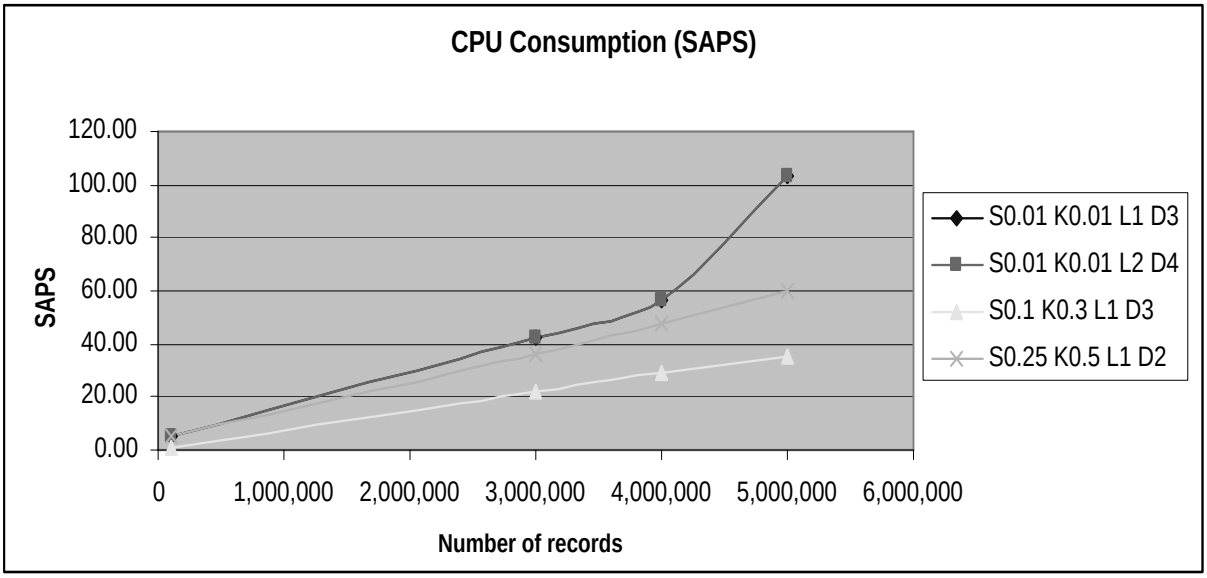
2,999,746	0.01	0.01	2	4	42.54	33.14	1122.30
3,993,689	0.01	0.01	2	4	56.58	43.37	1472.54
4,993,431	0.01	0.01	2	4	102.88	53.26	1790.38
99,979	0.1	0.3	1	3	0.53	0.97	21.92
2,999,746	0.1	0.3	1	3	21.63	25.99	715.69
3,993,689	0.1	0.3	1	3	28.86	33.93	935.55
4,993,431	0.1	0.3	1	3	35.31	41.54	1169.42
99,979	0.25	0.5	1	5	5.11	5.68	155.35
2,999,746	0.25	0.5	1	5	35.80	25.76	709.32
3,993,689	0.25	0.5	1	5	47.57	33.65	925.38
4,993,431	0.25	0.5	1	5	59.90	42.13	1156.77

As the above table shows with increase in the support and confidence value the CPU and DB usages go down. The results confirm that low value support and confidence result in more number of rules which increases both the CPU and DB usages. This also increases the memory consumptions

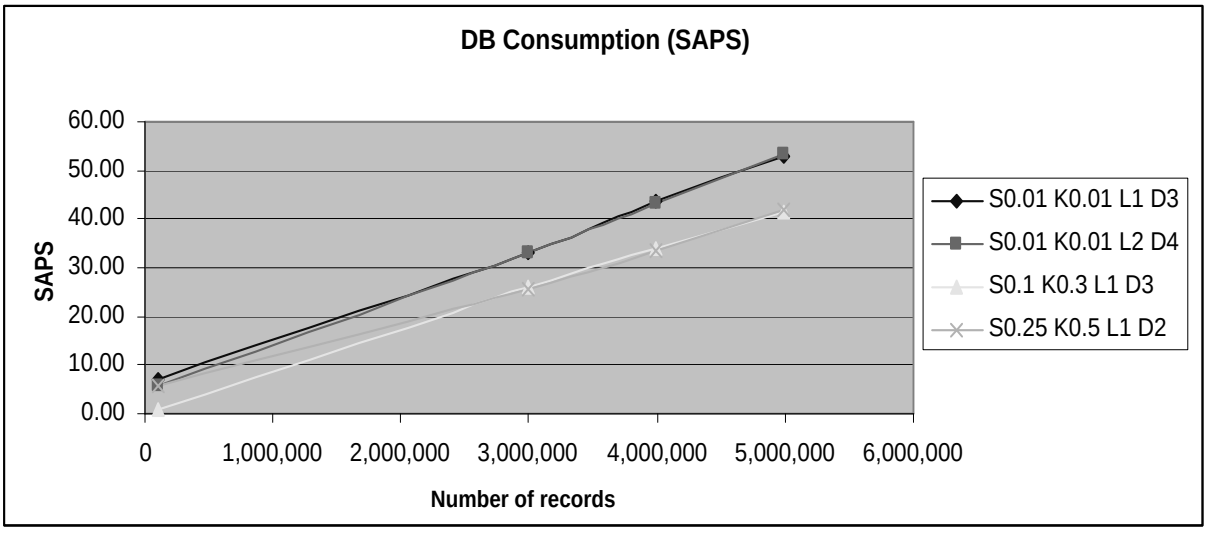
The next figure shows the memory consumption for the training of different models with varying number of input records. The results show that memory consumption significantly increases with low maximum support and confidence, as typically this leads to the identification of more rules and thus the size of intermediate results increases. For high number of input record the memory consumption may lead to several hundred megabytes. However, the results also confirm that due to the dynamic buffering mechanisms the slope of memory consumption decreases with increasing number of input records.



The next figure shows the CPU consumption for the training of different models with varying number of input records. The results show that CPU consumption s ignificantly increases with low maximum support and confidence, as typically this leads to the identification of more rules and thus the size of intermediate results increases.



The next figure shows the DB consumption for the training of different model s with varying number of input records. The results show that DB consumption increases with low maximum support and confidence, as this leads to the identification of more rules and thus the size of intermediate results increases.



where

S is minimum support

K is minimum confidence

L is number of maximum leading items

D is number of maximum dependent items

6 Sizing Figures of a Reference Site

Resource consumption for engine is dependent on many parameters such as the size, content of the data to be processed and mining model parameters. This section describes the resource consumed by each engine on internal reference test system. This allows estimating the resource consumption for each mining engine for a given parameters setting, and to derive according hardware sizing requirements.

6.1 Reference Site Details

The test system is a Hewlett Packard LH6000R with 4xP3-Xeon@700MHz/1MB Cache, 4GB RAM, 2SCSI Controllers with 8x36GB hard disks. This system configuration has a SAPS benchmark value of 1001.

6.2 Overview of resources required

Following table shows the overview of the resources required by all the engines.

Data Mining Engine	Number of Existing Model	Number of Training Models	Total No. Of Records Of All Models	Total Of Maximum Memory Consumption By All Models(MB)	Total Of Maximum CPU Consumption By All Models (SAPS)	Total Disk Space Consumption By All Models(MB)
Decision Tree	3	3	160000	113.16	240.38	90.10
Clustering	8	8	1320000	185.55	340.07	334.96
Scoring	4	4	660000	36.70	27.49	117.08
ABC Analysis	8	8	1320000	65.28	25	205.64
Association	4	4	12,086,845	1203.28	261.57	2488.37

7 Assignment of Resources to Data Mining Tasks

The execution of data mining tasks with mass-data in the range of gigabytes typically lead to long-running batch processes, bounding memory and CPU resources for a long time. The process management of the BW Web Application Server makes sure that these long-running batch processes do not have negative impact on parallel processes running on the same BW Web Application Server.

7.1 Memory Consumption

Although there is no mechanism to limit the resource consumption for data mining users or even data mining tasks, the Web Application Server provides basic mechanisms to limit the memory consumption of the application server and specific process types.

a) Server-specific memory limit. The Web Application Server provides parameters to limit the memory consumption of the Web Application Server. In case other applications like the DBMS or other Web Application Servers have to be executed in parallel on the same hardware server, this parameter allows balancing the memory consumption between the concurrently running processes. For more details about SAP Memory Management, please refer to the documentation of the SAP Web Application Server -> Client/Server Technology -> SAP Memory Management (SAP Memory Management link at SAP Help Portal).

b) Process-type specific memory limit. Every SAP Web Application Server provides a specific number of dialog work processes for the processing of online requests, and batch processes for the processing of batch requests like data mining tasks. A dispatcher dispatches the requests

coming from many applications, and assigns them to the dialog or batch processes for execution. This dispatching mechanism guarantees that there will be only limit number of parallel processes competing for CPU and memory resources, making sure that a long -running resource intensive batch process (like for data mining) will not lead to the crash or starvation of other processes running in parallel. The execution of data mining batch processes is treated like any other batch processes. In case a work process like a data mining process would exceed its upper memory limit, only this work process would be terminated without any impact on parallel running processes. To control the resource consumption of dialog work processes as compared to batch processes, the SAP Web Application Server allows the specification of upper memory limits for batch processes and dialog processes. Using these parameters batch processes could get more memory assigned than dialog processes and the other way around. However, the SAP Web Application Server does not allow the assignment of upper memory limits based on the requesting user or even the application/request to be executed. For more details about SAP Memory Management, please refer to the documentation of the SAP Web Application Server -> Client/Server Technology -> SAP Memory Management (SAP Memory Management link at SAP Help Portal).

7.2 CPU Consumption

The assignment of CPU resources to dialog and batch processes is controlled by the SAP Web Application Server and the operating system in a fair way, and there is no mechanism provided to influence this CPU scheduling based on the requesting user or even the application/request to be executed. Therefore, the assignment of data mining batch processes to the CPU is done in exactly the same way as for any other batch process.

7.3 Recommendations

Long-running data mining tasks “block” a batch work process for a long time, and therefore the throughput of parallel batch processes decreases during this time. If performance degradation of parallel processes in the BW system is a critical issue, the following settings can be applied:

a) Scheduling of data mining batch processes for times with low expected load. Data Mining batch processes can be started immediately or they can be scheduled for specific times. For example, a data mining batch process could be scheduled for the execution in the night window, thus guaranteeing that less dialog processes will be executed at this time. Note that the SAP Web Application Server allows the specification of different number of available batch and dialog processes for day and night window. Typically, customers are increasing the number of available batch work processes in the night and decrease it for the day, giving more resources to batch and dialog processes, respectively.

b) Dedicated SAP Web Application Server for data mining. A single BW Web Application Server can be installed which is dedicated to data mining. Data mining users would only log on to this specific Data Mining Web Application Server, and every data mining task would be executed on this machine. By using the parameters for upper memory limits described above, the maximum memory consumption of data mining processes can be controlled on this machine. Other critical BW tasks could either be executed on different Web Application Servers running on different hardware to avoid the competition for data mining resources at all, or they could be executed on different Web Application Servers running on the same hardware, but with guaranteed minimum resources due to the upper memory limits for the data mining Web Application Server. However, CPU scheduling of concurrently running Web Application Servers is up to the operating system and outside the control of the SAP Web Application Server.

8 Summary

8.1 Impact of CPU processing versus database access

The following table summarizes the general conditions for the impact of CPU processing and database access on the overall runtime of a mining method (CPU bound versus I/O bound).

Method	CPU Bound	I/O Bound (database bound)
Classification	In most cases, especially when no pruning is used	When pruning is used and tree complexity is low
Clustering	In most cases	
Weighted Table Scoring	In most cases, due to limited database access	
ABC Analysis	Quite Balanced (50%)	Quite Balanced (50%)
Association Analysis	Quite Balanced (60%)	Quite Balanced (40%)

8.2 Sizing recommendations

The results discussed in the previous sections have shown that the resource consumption highly depends on the data to be processed, the patterns to be discovered, and the model parameters specified. However, experiments and example-based estimations have shown the following:

- The runtime of most data mining methods are CPU bound, so the faster the CPU the better it is for the processing time of the algorithm. As the data mining algorithms do not use intra-process parallelism, having multiple CPUs in a BW server does not help in terms of CPU processing time for the data mining algorithm. However, as typically more than one process is running in a data warehouse, it is recommended to have one CPU dedicated for time critical data mining processes, and one or more additional CPUs available for concurrently running processes. Alternatively, if multiple single-CPU application servers are available for a BW, one application server could be dedicated to the execution of time-critical data mining runs.

  If the database management system and the BW web application server are running on the same server, then it is highly recommended to have at least two CPUs for the distribution of database and BW processing.

- Data mining, especially the training of association models based on tens or even hundreds of millions of transactional data records, may easily lead to several hundred megabytes or even few gigabytes of intermediate and permanent model result data to be stored in database tables. The table space of the database management system should be large enough to keep these dynamically generated result data.

- Main memory consumption, especially for the training of association models based on large amounts of transactional data, may lead to the consumption of several hundred megabytes. For an efficient execution of a data mining training maybe even under parallel load, it should be guaranteed that there is enough physical main memory available. As an SAP Web Application Server can manage up to 4 Gigabytes of memory, it is recommended to run time critical data mining training processes on a server with 4 Gigabyte of memory.

9 Comments and Feedback

Please send comments and feedback to Ramine Eskandari (ramine.eskandari@sap.com).

Appendices

Appendix A – Sizing of basic elements

- **How to estimate the size of an Extract Table (BW release < 3.5)**

  During training, prediction or evaluation of a data mining model, the query result gets fully written to the Extract Table. The number of records in the Extract Table is exactly the number of rows returned by the query. The size of a record in an Extract Table is given by the total of all fields (i.e. the size of the InfoObjects's data element) returned by the query. For example, if the query returns InfoObjects 0BPARTNER, 0REVENUE, and 0REGION , then with the help of the definition of these InfoObjects find the data element behind these InfoObjects (e.g. CHAR(10) which uses 10 Bytes, FLTP for floating point which occupies 8 Bytes, etc.). By summing up the sizes of these data elements for all InfoObjects returned by the query, and by summing the space needed by calculated key figures (the y do not have an InfoObject), the size of a single record in the extract table can be calculated.

  If the query read s all the fields and all records from an ODS, then the size of the extract table could be estimated by the size of the ODS. However, note that the ODS table may have additional metadata columns, and may use different representations of the data, so it is just an estimation.

  The input needed for a data mining model is defined by its mod el fields. Typically, a query should just return the fields needed for the data mining model. For example, an ABC model needs the field to identify the classified object (like partner number), the classification criterion (like sales), and grouping fields (optional). It is recommended to use a query just returning the fields needed . From a resource consumption point of view, it does not make sense to use a query returning say 10 characteristics and 10 keyfigures, when most of them are not mapped to the ABC model fields. The model fields do not have any impact on the extract table, as the extract table is exclusively determined by the query result.

- **How to estimate the size of Info Objects**

  To find the size or length of the given Info object, see the general tab page of the maintenance menu of transaction RSD1 and see the length field. (one can start the RSD1 transaction and provide the Info object name in the InfoObject field and press the Display button, details of the Info object including the length field is shown)

Appendix B – Recommendations and Limitations

- **Creating Query**

  The query should be designed in a way that they contain only the fields required by the model. This ensures the size of extract tables created would be smaller.

- **Limitations**

  The number of characteristics in a query used as input f or data mining is limited to 16. An ODS can have a maximum of 16 key fields and a maximum total number of 740 fields.

Glossary

- **APD**

  The Analysis Process Designer (APD) is a workbench with an intuitive visua l interface that enables one to visualize, transform, and deploy one's data from business warehouse. It combines all these different steps into a single data process that one can easily interact with.

- **SAPS** Based on customer requests, SAP has defined a un it for measuring throughput of mySAP.com. The unit is:

  SAPS: SAP Application Benchmark Performance Standard

  100 SAPS are defined as 2,000 fully business processed order line items per hour in the standard SD application benchmark. This throughput is achiev ed by processing 6,000 dialog steps (screen changes) and 2,000 postings per hour in the SD benchmark or processing 2,400 SAP transactions. Fully business processed in the SD standard benchmark means the full business workflow of an order line item (creatin g the order, creating a delivery note for this order, displaying the order, changing the delivery, posting a goods issue, listing orders and creating an invoice) is processed.

- **InfoCube**

  An object that can function as both a data target and an InfoProvider. From a reporting point of view, an InfoCube describes a self-contained dataset, for example, of a business-orientated area. An InfoCube is a quantity of relational tables arranged according to the star schema: A large fact table in the middle surrounded by several dimension tables.

- **InfoObject**

  Business evaluation objects are known in BW as InfoObjects. They are divide into characteristics (for example, customers), key figures (for example, revenue), units (for example, currency, amount unit), time characteri stics (for example, fiscal year) and technical characteristics (for example, request number).

- **ODS**

  An ODS object acts as a storage location for consolidated and cleaned -up transaction data (transaction data or master data, for example) on the document (atom ic) level. An ODS object contains key fields (for example, document number/item) and data fields that can also contain character fields (for example, order status, customer) as key figures. The data from an ODS object can be updated with a delta update int o InfoCubes and/or other ODS objects or master data tables (attributes or texts) in the same system or across different systems.

- **Model**

  Model is metadata definition based on which a data mining method is executed. Model contains information of data fields that will be used and other method specific parameter values.