Delivering information you can trust
July 2008

IBM Information Management software

The seven essential elements to achieve highest performance & scalability in information integration

Contents

Executive summary

Every day, torrents of data inundate IT organizations and overwhelm the business managers who must sift through it all to glean insights that help them grow revenues and optimize profits. Yet, after investing hundreds of millions of dollars into new enterprise resource planning (ERP), customer relationship management (CRM), supply chain management (SCM), business intelligence (BI), business process management and data warehousing systems, many companies are still plagued with disconnected, "dysfunctional" data—a massive, expensive sprawl of disparate silos and unconnected, redundant systems that fail to deliver the desired single view of the business.

In order to meet the business imperative for enterprise integration and stay competitive, companies must manage the increasing variety, volume and velocity of new data pouring into their systems from an ever-expanding number of sources. They need to bring all of their corporate data together, and deliver it to end-users as quickly as possible in order to maximize its value. And they need to integrate data at a more granular level – dealing at the individual transaction level rather than with general summary data.

To address these challenges, organizations require a scalable information integration architecture that has:

1. *A dataflow architecture supporting data pipelining that allows data to process from input to output without landing to disk, for a variety of operations such as profiling, cleansing and transformations*
2. *Dynamic data partitioning and in-flight repartitioning of data*
3. *Scalable hardware environments, portable across SMP, clustered environments, and MPP platforms that don't require modifications of the data flow design*
4. *Support for leading parallel databases including IBM® DB2® UDB, Oracle, and Teradata in parallel and partitioned configurations*

Benefits for CRM, Operations, and Sales

- ***One telecommunication company is able to increase the number of marketing campaigns it executes in the long distance market from 4 to 40 per month because of a parallel infrastructure, allowing them to become profitable 18 months ahead of projections.***
- ***One transportation company created a yield management application that allows them to re-price their service as often as four times per day, creating $100 million in incremental revenue annually.***
- ***One bank could only get summary data out of its data warehouse, but was not able to perform high-value analysis off of summary-level data. With a parallel infrastructure that allows it to analyze more granular customer transaction data and parallelize SAS, it expects to generate $100 million in additional revenue annually.***

5. *High performance & scalability not only for bulk / batch movement but also for real-time data processing*
6. *Extensive tooling to support resource estimation, performance analysis and optimization*
7. *An extensible framework to incorporate in-house and third-party software*

The architecture must be able to grow with the organization as data volumes grow and performance requirements increase. Some of the most important success criteria for an architecture's scalability are:

- *The existence of any upper bounds – i.e. are there any limits when additional resources do not lead to improved performance,*
- *Linear (or better) performance improvements when adding hardware resources – i.e. will n additional resources lead to n-times better performance (n = 2, 4, 8, …), and*
- *Minimal non-hardware related costs when the environment changes due to changes in data characteristics or hardware resources are added. In order to achieve best return-on-investment for your data integration project, it is critical to consider the overhead in labor-intensive configuration changes when the environment changes. Adding processor(s) or nodes to the hardware environment should occur with no change to the design of your data transformations and the end-to-end flow in order to avoid re-testing, re-compiling and deploying.*

The case for parallelism

This paper is written for business and technical decision-makers responsible for designing, building, supporting, and using scalable data processing systems. A combination of factors is fueling the dramatic growth of all forms of digitized data. In order to compete, organizations need a more granular level of detail (data) – individual transactions, not just summary data – and they need it faster in order to respond to changing market and competitive pressures. Consider these examples:

- *In order to make faster business decisions, one large retail organization with nearly 2,000 stores in North America wants to collect transactional data as it occurs – every 15 minutes – with potentially hundreds of transactions per hour in each store; the data volume and performance requirements are enormous.*
- *Currency trading is a 24 hour a day business; brokerage houses need to provide this data to traders in real-time to react to shifts in the market.*
- *A telecommunications company is increasing the number of US states where they are offering long distance service from 2 to 14 in one year; the data volume and processing requirements for their marketing campaigns' data warehouse will greatly increase during that time.*

To support this growing data volume, variety, & velocity, and the transitions from monthly or weekly batch runs to daily or up to the minute data transitions, builders and users of enterprise data warehouses require a high performance and scalable architecture. Beware however: Not all "high performance" architectures are alike. Terms such as "parallel processing" and "scalability" carry different connotations and meanings from different vendors, analysts, and industry experts. This paper explains the seven key elements that IT organizations must consider when evaluating the real capabilities of a high performance and scalable data infrastructure solution. These seven key elements for a parallel architecture are:

1. *A data flow architecture supporting data pipelining without the necessity for landing data to disk,*
2. *Dynamic data partitioning and in-flight repartitioning of data*
3. *Design once, deploy flexibly and achieve scalability on a variety of hardware environments,*
4. *Support for parallel access to parallel databases,*
5. *Integrated platform for bulk / batch movement as well as real-time / trickle-feed processing,*
6. *Extensive tooling for resource estimation, performance analysis and optimization,*
7. *An extensible framework to incorporate in-house and third-party software.*

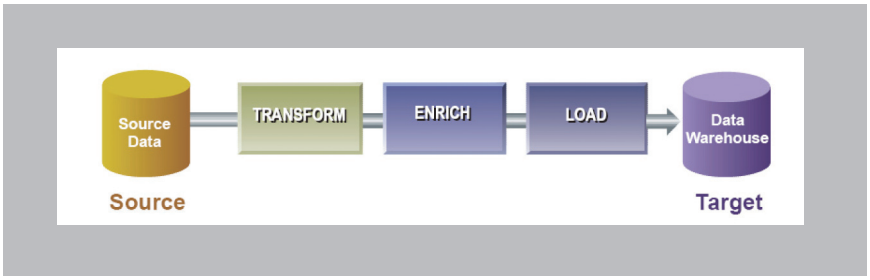
1. Data flow architecture supporting data pipelining

In considering the key issues associated with global, highly scalable enterprise data warehousing applications, IT and data management staff typically wish to accomplish many steps in a flow – picking up data from source machines, transforming the data, enriching, and ultimately moving the data to the enterprise data warehouse or other systems such as data marts or OLAP tools – while at the same time minimizing or eliminating any costly access to disk storage between steps.

IT development organizations should demand an information integration platform and parallel-processing framework based on the data-flow model which allows developers to create visually a sequence of operations that can effectively manipulate data for quality and transformation purposes.

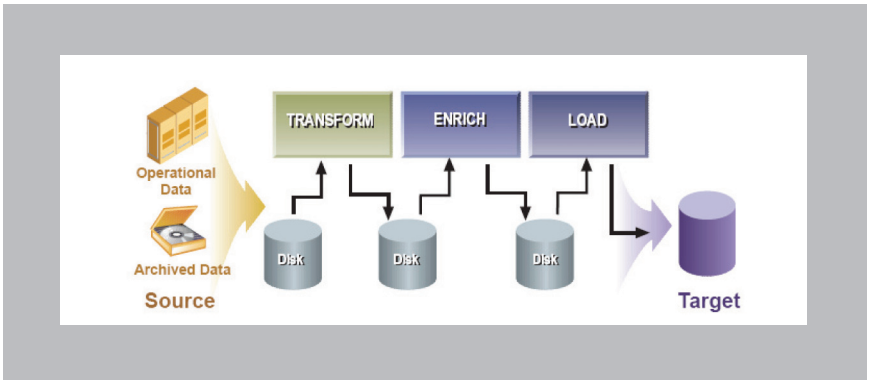
Data can be coming in from multiple data sources, such as flat files, databases, packaged applications (like SAP, JD Edwards, etc.), or as streams in real time. In all these cases, high throughput based on a data flow architecture remains important.

Figure 1: Data Flow Architecture



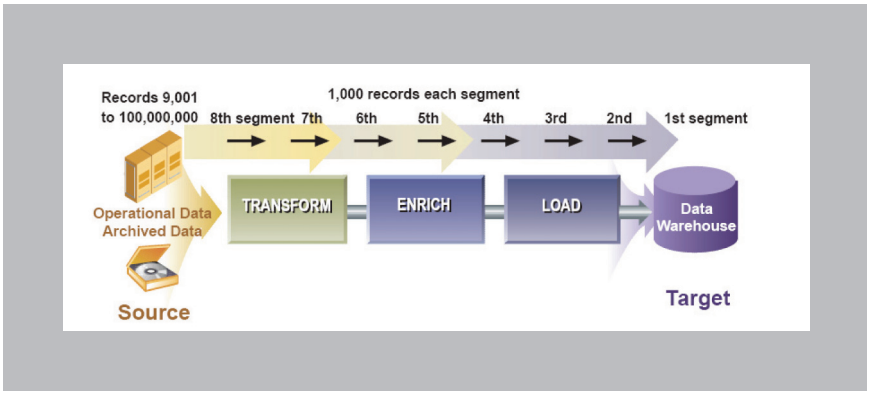
Traditional information integration approaches typically run all the data through an individual step, and generally write the data to disk, before starting the next step in the application. This creates a start-stop-start sequence that bogs down the application and severely reduces performance. This also creates an inordinate amount of disk usage – one execution of a simple job can easily use 4 to 7 times the disk space as the original source data occupied, creating a disk management nightmare. It quickly becomes impractical for large data volumes – disk I/O consumes the processing and terabytes of disk are required for temporary storage.

Figure 2: Traditional batch processing



Shown in Figure 3, data pipelining eliminates the incremental writing and reading to disk by flowing data from upstream processes immediately to downstream processes when it is available using shared memory and piping, even before the upstream process completes. Data pipelining also optimizes the distribution of load among available resources "horizontally" (from source to target): while upstream operations (on one node) still process data but start to produce results, downstream operations (on another node) can start their processing as soon data arrives.

Figure 3: Data Pipelining



To be more precise, data is (or can be) buffered in blocks so that each process is not thrashing the system when executing one component or the next. This avoids deadlocks and greatly accelerates performance by allowing both upstream and downstream processes to execute concurrently.

Without a data flow architecture that supports data pipelining, the implications are that:

- *Data must be landed to disk between each process, severely degrading performance, greatly increasing storage requirements, and creating a disk management nightmare*
- *The developer must manage the I/O processing between each component*
- *The process becomes impractical for large data volumes.*

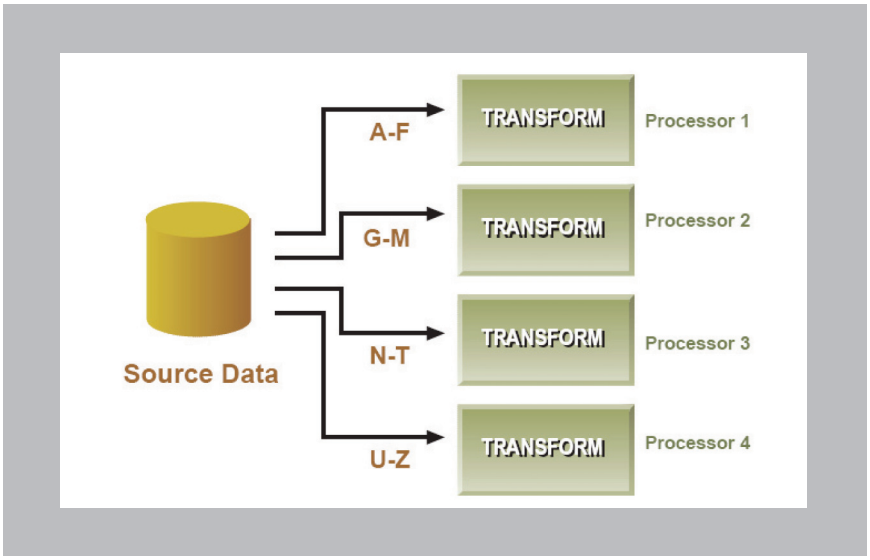
The application will be slower, as disk use, management, and design complexities increase.

2. Dynamic data partitioning, in-flight data re-partitioning

As described above, data pipelining is one approach to improve performance and in particular to eliminate intermediate data staging. Multiple operations in the end-to-end data flow sequence work "horizontally" (along the data flow sequence) in parallel on multiple nodes. Although data pipelining can greatly improve performance, it has also its limitations, in particular in the beginning and towards the end of processing a data flow: Some downstream operations need to wait until the first data entries "trickle" through the flow and some upstream operations will be idle after they have completed their processing while downstream operations finishing.

Data partitioning is a complimentary approach to achieve parallelism which distributes the load "vertically" among multiple instances of the data flow against separate data partitions. The selected scope of source data is split into sub sets which are called partitions. Multiple instances of end-to-end data flows which contain a sequence of operations then process the partition that is assigned to that instance.

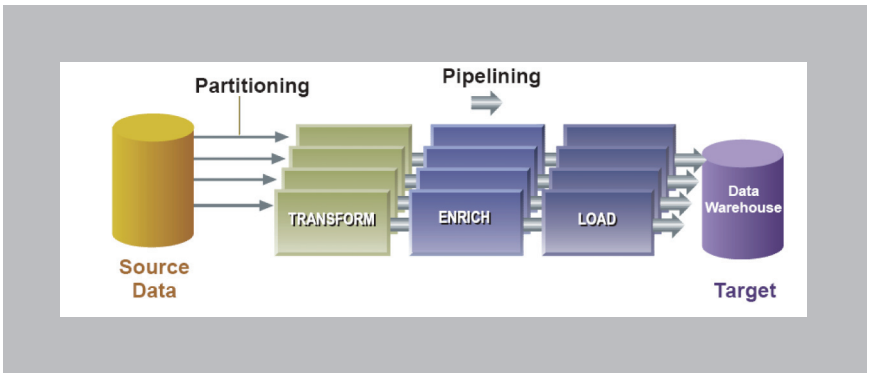
Figure 4: Partitioning customer names



Data partitioning is well suited to many commercial data-processing applications because data records can usually be partitioned along a single variable (for example, customer account number, zip code, or transaction date) and thereby benefit from the parallel execution of application logic. Figure 4 shows data partitioning of customer names beginning with A-F executing in one partition (processor), G-M in another, and so on.

Figure 5 shows an example of parallelism achieved through executing multiple instances of application logic against partitioned data.

Figure 5: Data Partitioning and Parallel Execution



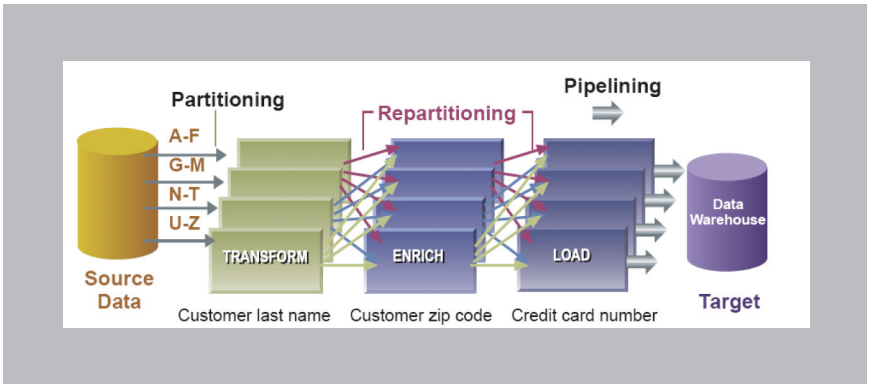
A scalable architecture should support many types of data partitioning, including:

- *Key (data) values*
- *Range*
- *Round-robin*
- *Random*
- *Entire*
- *Modulus*
- *Database matching partitioning (e.g., DB2)*

One key characteristic of those partitioning mechanisms is that they distribute the source data automatically in balanced partitions so that each partition has approximately the same number of entries.
Typical information integration tools lack this capability and require developers to "hard-wire" data partitions. When using these tools, architects or administrators must manually assign boundaries of partitions, e.g. by specifying the value that represents the boundary. This method is highly inefficient and results in costly and time-consuming rewriting of data flows or the data partitions whenever hardware capacity or source data volume/characteristic changes. This can consume many weeks or months of development and testing prior to production. Even worse, this labor-intensive effort has to be frequently repeated in dynamic environments.

Keep in mind that the developer should not have to be concerned about the number of partitions that will execute, the ability to increase the number of partitions, and more importantly, data re-partitioning.

Figure 6: In-flight Data Re-partitioning



In the example above, data was partitioned based on customer last name and then the data partitioning was maintained throughout the flow. This is impractical for many uses. Consider a transformation that is based on customer last name, but the enriching needs to occur on zip code – for house-holding purposes – and then loading into the warehouse is based on customer credit card number (more on parallel database interfaces below). With *in-flight* or *dynamic data re-partitioning*, data is re-partitioned between processes based on the downstream process data partitioning needed on-the-fly, without landing the data to disk. Meaning, this is done in memory. Keep in mind that data is also being pipelined to downstream processes when it is available.

Most information integration tools can not dynamically repartition data; they require separate manual "mappings" for each process, which forces data to disk multiple times in between steps in order to complete each data flow. Depending on the process and size of data, these I/O delays could increase processing times by anywhere from 2 to 10 times or more.

The implication without partitioning and in-flight data re-partitioning is that the developer must:

- *Create separate flows for each data partition, based on the current hardware configuration*
- *Land data to disk between processes*
- *Manually re-partition the data*
- *Start the next process*

Consequently, the resulting application will be slower, use more disk and disk management, and have greatly increased design complexity.

3. Highest scalability across a variety of hardware environments at minimal cost

Hardware vendors have offered scalable parallel computers for many years. Computing architectures span small, single processor machines, multi-CPU systems, giant clusters, and systems that have dedicated memory and disks. First, some definitions:

Uniprocessor

A uniprocessor machine has dedicated memory and disk for its single CPU. Examples include PCs, workstations, and single processor servers.

Symmetric multiprocessor (SMP)

A Symmetric Multiprocessor (SMP) system is a multiple-processor environment that shares everything – memory and disk – across the CPUs.

Clustered and massively parallel processor (MPP)

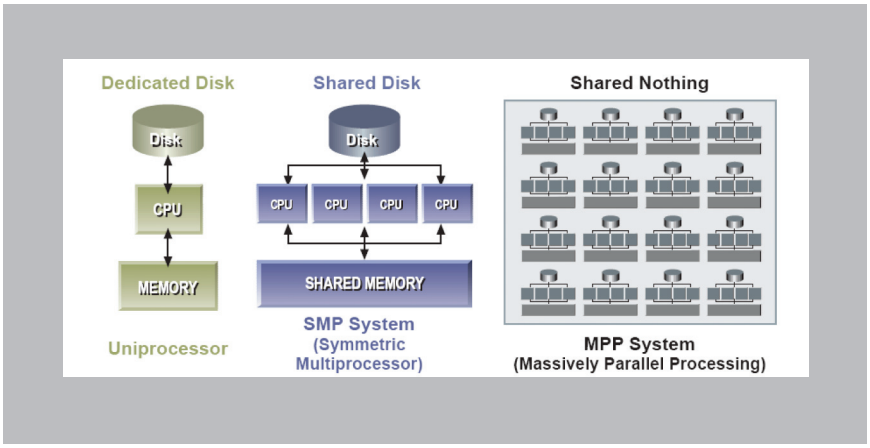
Clustered environments and Massively Parallel Processor (MPP) systems are shared nothing environments. Each CPU or node (a single CPU or SMP) has dedicated memory. Clusters often have SAN-based shared storage. An MPP is a cluster without shared storage. Clusters and MPP environments can have 2 to hundreds of processors.

Grid computing

With the commoditization of hardware computing power, grid computing is becoming a highly compelling option for large enterprises. Grid computing allows companies to bring more processing power to bear on a given task than ever before possible.

Grid computing takes advantage of all distributed computing resources – processor and memory – available on the network to create a single system image. Grid computing software provides a list of available computing resources and a list of tasks. When a machine becomes available, it assigns new tasks according to appropriate rules. There can literally be thousands of machines available on the grid. What grid-computing software does best – balancing IT supply and demand by letting users specify their jobs' CPU and memory requirements, then finding available machines on a network to meet those specs – isn't necessarily an advantage for business-computing tasks such as managing the flow of raw materials and finished goods in a supply chain or selling products through an E-commerce Web site. Grid computing provides a set of horizontal integration capabilities that effectively addresses the challenge of cross-enterprise, cross-functional, IT resource integration and even extends that solution among multiple organizations. Grid computing is good news for batch throughput; however it is unlikely to replace big symmetric multiprocessing systems for running applications dependent on serial logic and large databases, such as those from Oracle and SAP.

Figure 7: Hardware environments



Enterprise data warehouses must not only be able to support the range of hardware architectures, but more importantly, accommodate growth as data volumes and complexity increases.

In order to make the best use of development resources, optimize use of hardware, and avoid hitting performance walls down the road, IT organizations should demand that information integration applications developed on a workstation can run, without recompilation, on that workstation, on an SMP server, or on a large scalable cluster or MPP system. Even more importantly, the information integration approach should not force developers to change data flow designs because of a change in database characteristics or when hardware resources are added.

The key to this is a clear separation between the expression of the data flow logic (developer's responsibility) and the mapping of that logic to the underlying parallel hardware platform (data integration software's responsibility).

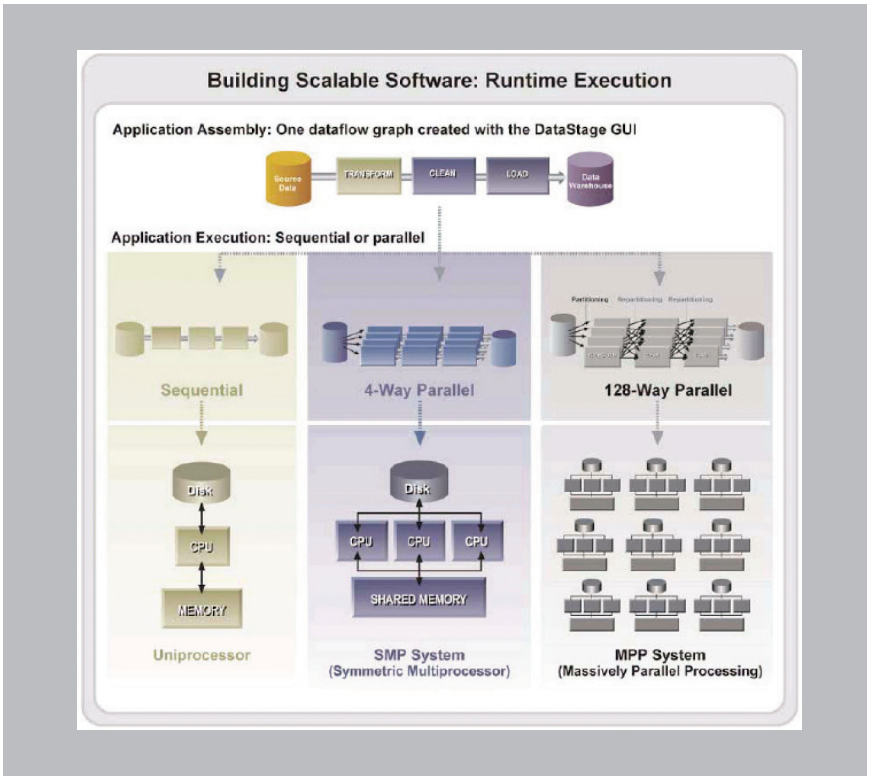


Figure 8: Putting it all together: Dataflow to partitioning & dynamic repartitioning to hardware deployment

Some data integration vendors claim they can run on SMPs and MPPs. Although they do allow to deploy data flows on different hardware configurations, there are some very important differences.

The first key differentiating advantage of InfoSphere Information Server is that you design your data flow once (without considerations of how many processors you will deploy it on), and then deploy flexibly. You can deploy the same job without any changes to the job on a uni-processor, SMP, MPP, or GRID environment. Traditional information integration tools require much more labor-intensive changes: you may need to manually assign individual operations to run on a dedicated processor, or even worse to manually redesign your data flow and to split it up in multiple "sub" data flows. Whereas you only register new nodes in the configuration file of InfoSphere Information Server – without even recompiling the data flows – other traditional information integration tools require labor-intensive changes and most likely a series of compile-test-tune-deploy cycles.

Also, is the data integration platform truly saturating all of the nodes of the MPP box or systems in the cluster/grid? Also equally important is letting the data integration infrastructure optimize the use of all of the available hardware resources. If, for example, a user wanted to run a project in parallel on 4 processors during the daytime and then in 20-way parallel mode at night when additional resources are available, this would actually require extensive rewriting of the data flow jobs. If the data integration software does not seamlessly handle this, then it must be manually done by development. This means you can't maximize available hardware and spare computing power, nor can you easily scale as performance needs increase. Most information integration approaches cannot dynamically adjust to a change in the environment and to re-balance the load automatically. Newly added hardware resources remain idle because a complex data flow cannot be automatically and transparently to the user divided into smaller components that are then delegated to the additional resources. Changes in data volume and characteristics also negatively impact the overall performance in most other products. The system doesn't automatically re-partition the load to re-balance the overall workload. Instead, most other information integration platforms force the developer

- *to recognize the change,*
- *to know how to change the design to better balance the load,*
- *to make that change,*
- *to test the modified design,*
- *to deploy it,*
- *then to analyze the performance in the production environment,*
- *and most likely to iterate through this change process again and multiple times until the workload is optimized (until it changes again).*

Obviously, this can be a time consuming and therefore expensive undertaking. Even worse, many customers have a dynamic environment where such changes will occur frequently.

Without support for scalable hardware environments, the implications are:

- *Slower execution because all available hardware resources are not maximized*
- *No decoupling of application design and hardware configuration which will require manual intervention for every hardware change*
- *An inability to scale on-demand.*

4. Parallel database connections

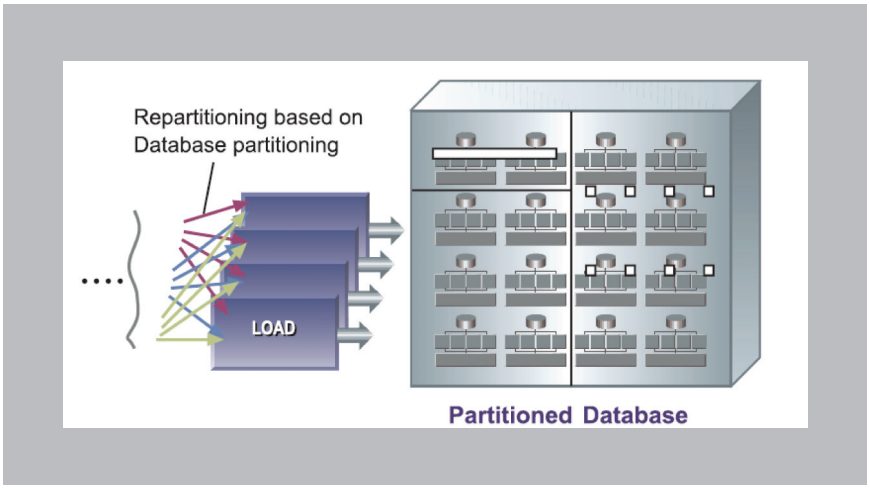
Enterprises which have parallel hardware and parallel relational databases are often unable to realize all of the benefits of end-to-end parallelism because their information integration software does not allow users to extract or load data in parallel from the database. This situation creates bottlenecks, undermines true scalability and leaves IT organizations to cope with just a single connection between the relational data and the application. This inefficiency often causes batch processing windows to balloon.

Many relational Database Management Systems, such as DB2 UDB, support partitioning of a database within a single server or across a cluster of servers. This capability provides multiple benefits including scalability to support very large databases or complex workloads and increased parallelism for administration tasks.

A true parallel processing infrastructure should support parallel access to leading databases – IBM DB2 UDB, Oracle, Informix, and Teradata – automatically. Productized database interfaces should support pulling and pushing multiple data streams in and out of the database – as well as running transaction logic – all in parallel to avoid any sequential bottlenecks in processing. In addition, data partitioning should be done consistent with how the database partitions the data (across nodes).

Figure 9 shows data being re-partitioned before calling the Load operation. The Load process, running in parallel, uses the database load interface or utility to load the database into the database partitions. These partitions could be across clusters or nodes. The converse should also be true, that being unloading or extracting in parallel based on the partitioning of the database.

Figure 9: Re-partitioning based on Database Partitioning



Even files should be able to read in parallel. Each partition should be able to read a contiguous range of records from the input data file. The other partitions should know what records to read in its partition. The resulting data set contains one partition per instance of the file read operation.

Many information integration tools simply do not support parallel loading nor do they support automatic re-partitioning of data based on the source or target database partitioning. With this seamless parallel extraction and loading, developers can much more readily focus on information integration tasks and avoid dealing with complexities of the database.

Without supporting parallel database interfaces and database partitioning, the implications are:

- *That extraction or loading will bottleneck into a single, sequential process greatly slowing performance minimizing the advantage of using a parallel database*
- *This will force data to disk in order to re-partition data before the load process making the flow slower*
- *Developers will have to handle the complexities of the parallel database connections and re-partitioning*

In summary, the application will be slower, have increased disk use and management, and greatly increased design complexity.

5. Real-time processing and change data capture

Data transformation has involved from batch and bulk data movement to also include real-time data transfer based on change data capture (CDC). Whereas batch and bulk data movement is scheduled on a relatively infrequent basis for all data, real-time data transformation occurs whenever the data at the source changes for just the data that is changed. The change data is captured, transferred and transformed and then loaded into the target.

One important factor influencing performance and scalability in real-time data transformation is the model to capture a change at a source. One option that is adopted by some technology providers is for the data transformation engine to "pull" the source for any changes since the last pull request. Although this can be easily implemented, it has negative implications on performance because the transformation engine has to ask for any change instead of just receiving the change. The second option is for a change data capture mechanism to "push" changes as data streams. As soon as data is modified at the source the CDC mechanism becomes aware of the change and forwards the modified data to be further transformed and processed. InfoSphere Change Data Capture provides the second, more efficient option.

There is a spectrum of options to capture a change in a source before pushing or publishing the change for further processing. It ranges from simple trigger-based mechanisms to highly advanced log scraping technologies. The advantage of a log-based capture approach that is implemented by InfoSphere Change Data Capture is the lower impact to the source database which results ultimately in higher performance of the overall approach. Instead of putting the burden of identifying the change on the database engine – e.g. when using triggers, a dedicated, small-footprint CDC technology reads the changes directly from the database log file.

The third important aspect of a CDC technology is whether or not data needs to be temporarily persisted between capturing the change and processing it during data transformation and load into the target. One of the unique advantages of InfoSphere Change Data Capture is that changes can be streamed without persisting them along the way. This increases performance further since the data does not need to be written to disk and then accessed from disk by a data transformation engine.

6. Tooling for performance analysis and optimization

The first important step to ensure high performance is to have sufficient resources for the required task. Insufficient resources (CPUs, disk, etc) have a significant impact on the overall performance. The architect and developer need to understand before they deploy the data transformation process in a production environment what the required resources will be and to identify bottlenecks ahead of time. The resource estimation can increase performance significantly, and even more important avoid multiple cycles of deploying a data flow on insufficient hardware, adding more resources, redeploying it, testing the impact, etc. The best solution to a problem is not to have the problem. And efficient tooling that can simulate a test run with parameters such as a specified data volume estimates the required resources and is therefore a critical component to ensure performance and scalability.

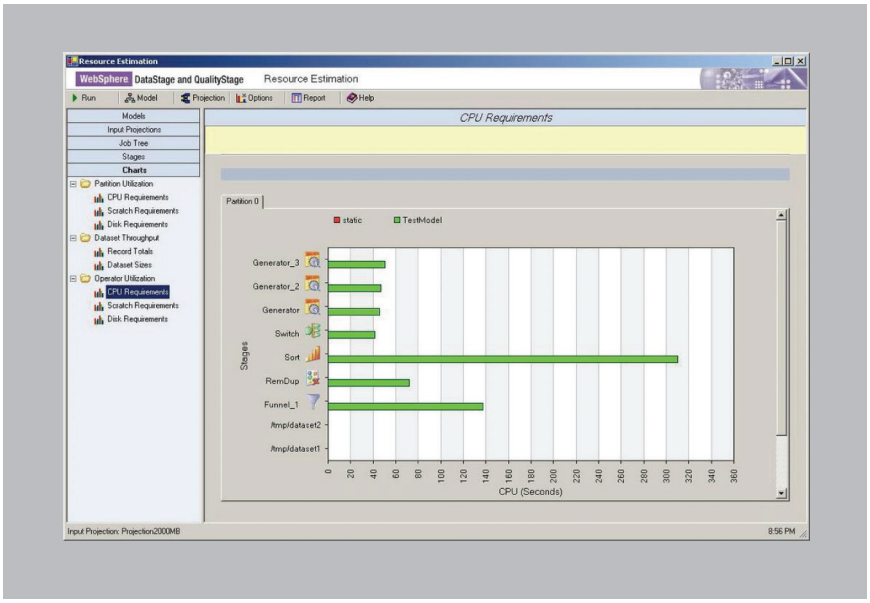
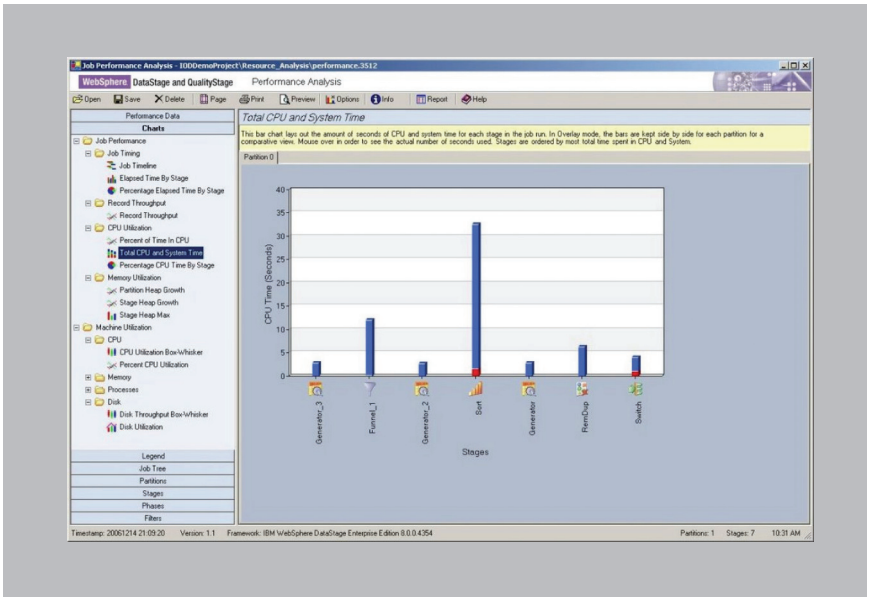


Figure 10: Resource Estimation

Even if the required resources are perfectly planned ahead of time, it is important for the architect and developer to understand how well data transformation jobs are executing: how long did a job take, what was the elapsed time by individual stages (i.e. transformation operations), what was the record throughput, what was the CPU utilization of individual stages, what was the memory utilization, etc. Graphical representation to very detailed statistics on job execution (with drill down to partitions and stages) and resource utilization makes it easy for the architect and developer to quickly assess problems and take necessary actions to guarantee highest levels of performance.

Figure 11: Performance Analysis



7. Beyond ETL…enabling third party software

A scalable infrastructure should provide native, high-performance parallel components, in particular sorting, aggregation, joins, and so on. But because any large enterprise has special and customized needs, a scalable infrastructure should be extensible in order to integrate existing programs and third-party tools as part of the information integration process. These programs originally written to execute sequentially – should be able to execute in parallel on a partition of data, and regardless of the programming language used (C, C++, COBOL, etc.). A key requirement in order to integrate existing software code is the ability to only operate on the data (columns/fields) of each record and for the infrastructure to simply pass the rest of the data not used (touched/changed) through the component to the next downstream component in the data flow. This has been referred to as column or schema propagation. This is a critical aspect in order to integrate existing applications without change, making them more portable and useable. With this ability, software can be integrated and parallelized.

Third-party tools should also be able to be integrated and executed in parallel, including SAS. Many vendors claim to integrate with existing and third-party tools. They usually do this by landing data to disk, then calling the "external" program. This is usually done through manually writing scripts – which is not an integrated solution and definitely not executing them in parallel. Keep in mind a truly scalable architecture incorporates these non-native components and tools taking advantage of data partitioning, dynamic re-partitioning, and pipelining all without landing data to disk between operations on any hardware environment.

Without an extensive framework that can incorporate existing programs and third-party applications:

- *Can not be integrated into the data flow*
- *Requires data to be collected together back into one stream from its partitions and landed to disk*
- *Manually invoke the program sequentially*
- *Re-start the next flow and partition the data*

The application will be slower, use more disk and disk management, require manual intervention or script writing and thus greatly increase design complexity.

A word about high performance sorting…
Since sorting data is typically a critical and time-intensive task in any large-scale information integration effort, IT organizations should ensure that parallel infrastructure software has a built-in high-performance sort to sort the records of a data set. In the absence of this, sorting operations can create unacceptable time delays and processing bottlenecks. To accommodate high data volumes, this sorting operation should be capable of running on a single processor to sort an entire data set or on multiple processors to sort the records in each partition of a data set – all without landing to disk and incurring the associated I/O performance degradation. When coupled with an appropriate range partitioner, a partition sort operation produces a completely ordered data set in which the records in each partition are ordered and the partitions themselves are ordered.

Customer case study – InfoSphere Information Server boost throughput at MGM Mirage by 10 times

Guest satisfaction is paramount to MGM MIRAGE, and the company needed a single view of customers to improve customer service, ensure proper recordkeeping and accounting, enable targeted marketing programs and provide a critical basis for both tactical and strategic decisions. The single view needed to encompass customer data from its multiple casinos, hotels and other sources, including the newly launched Players Club loyalty program.

MGM MIRAGE had designed a warehouse to house the data that would form the basis for its single customer view, but the company was seriously constrained by its technology architecture. The existing environment consisted of a series of extract, transform and load processes that had been developed by a local integrator. MGM MIRAGE technical staff had no access to the source code, and thus was unable to make modifications to the programs without ongoing reliance on the third party. The result was high costs.

With the IBM InfoSphere Information Server software, MGM MIRAGE has the ability to access real-time customer information, providing the ability to analyze customer behaviors for marketing and other purposes. This also includes the ability to pull data from a variety of sources including the new Players Club loyalty program, the hotel and table games.

The enterprise data warehouse match/survive processes have given MGM MIRAGE the ability to cross-market key promotions to properties not yet included in the Players Club loyalty system, thus bringing added revenue at an accelerated pace.

Benefits such as improved customer service, reduced costs for customer mailings and an improved basis for decision-making resulted as well. With the InfoSphere Information Server SOA solution, MGM Mirage is creating on-demand integration environment, with an up-to-date and authoritative customer database. With same-day visibility into customer visits and gaming activity across all its Las Vegas and non-Las Vegas casinos through the company's new customer data warehouse and CRM platform, MGM Mirage has unprecedented ability to profile and promote to its most profitable and active gambling customers.

In addition, MGM MIRAGE took preemptive steps to handle increased load by creating an EDW grid architecture based on Linux. The installation of the grid project was completed in only six business days and test jobs were deployed within only one day. The grid dramatically improved run statistics. For example, performance for a test job that would traditionally take 1 hour and 40 minutes was reduced to only 12 minutes on the grid. Another test project that would take 33 hours on the non-grid architecture was reduced to only five hours on the grid.

With IBM InfoSphere Information Server software, MGM MIRAGE will meet its critical objective of self-sufficiency in managing its data warehouse environment, eliminating reliance on a third party for both code development and code maintenance and significantly reducing costs for creating and maintaining integration solutions.

In addition, the IBM implementation will result in a streamlined environment that MGM MIRAGE can control and modify as new requirements emerge, for improved agility and flexibility as well as cost savings. Not only will the company be able to create and maintain data movement jobs efficiently with the WebSphere® DataStage® software, but also it will enjoy the benefits of higher-quality data. Benefits such as improved customer service, reduced costs for customer mailings and an improved basis for decision making will result from the implementation of the InfoSphere Information Analyzer and InfoSphere QualityStage® software.

With the InfoSphere SOA solution, MGM MIRAGE will be able to create an on-demand integration environment, with an up-to-date and authoritative customer database. With same-day visibility into customer visits and gaming activity across all its Las Vegas and non-Las Vegas casinos through the company's new customer data warehouse and CRM platform, MGM MIRAGE will have unprecedented ability to profile and promote to its most profitable and active gambling customers.

IBM InfoSphere DataStage provides world-class data transformation capabilities, delivers new metadata enhancements designed to improve developer productivity and enable fast data flow design for creating and populating data warehouses. The latest version of the product also achieves unprecedented performance and scalability.

IBM InfoSphere QualityStage is newly redesigned to deliver massive productivity gains and control over integration. It leverages the "design-as-you-think" paradigm of the InfoSphere DataStage user interface, enables a new class of user to create quality rules, and provides greater developer productivity with interactive visual design of data quality rules and instant feedback to allow developers more control over fine-tuning of quality logic.

Summary

In this paper, we have introduced seven elements that are critical to ensure the highest degree of performance and scalability of your information integration deployment.

1. *A dataflow architecture supporting data pipelining that allows data to process from input to output without landing to disk, for a variety of operations such as profiling, cleansing and transformations,*
2. *Dynamic data partitioning and in-flight repartitioning of data*
3. *Design once, deploy flexibly and achieve scalability on a variety of hardware environments at minimal costs, portable across SMP, clustered environments, and MPP platforms*
4. *Support for leading parallel databases including IBM DB2 UDB, Oracle, and Teradata in parallel and partitioned configurations*
5. *High performance & scalability not only for bulk / batch movement but also for real-time data processing,*
6. *Extensive tooling to support resource estimation, performance analysis and optimization,*
7. *An extensible framework to incorporate in-house and third-party software*

We have highlighted the unique advantages that InfoSphere Information Server delivers as the best information integration platform to achieve the highest level of performance and scalability and to ensure best ROI.

For more information

To explore how IBM can help your organization realize the promise of information grids—gaining a single view of critical data—contact your local IBM sales account executive or visit:

IBM Information Integration: **ibm.com**/software/data/integration
IBM InfoSphere DataStage: **ibm.com**/software/data/integration/datastage
IBM Grid Computing: **ibm.com**/grid
IBM BladeCenter: **ibm.com**/systems/bladecenter

IBM Software Group
Route 100
Somers, NY 10589

Printed in the United States
July 2008

TAKE BACK CONTROL WITH **Information Management**