Seven principles for achieving high performance and scalability for information integration
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), master data management systems (MDM), business intelligence (BI) data warehousing systems or big data environments, 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.

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 their corporate data together, deliver it to end users as quickly as possible to maximize its value, and integrate it at a more granular level than ever before—focusing on the individual transaction level, rather than on general summary data. As data volumes continue to explode, clients must take advantage of a fully scalable information integration architecture that supports any type of data integration technique such as extract, transfer and load (ETL), data replication or data virtualization.

Such scalable information integration architecture must have the following qualities:

1. A data flow architecture supporting data pipelining that allows data to be processed from input to output without landing to disk as data is moved between different operations, such as profiling, cleansing and transformations
2. Dynamic data partitioning and in-flight repartitioning of data
3. Scalable hardware environments, portable across symmetric multiprocessing (SMP) clustered environments and massively parallel processing (MPP) platforms, that do not require modifications of the data flow design
4. Support for leading parallel databases, including IBM® DB2® Universal Database (UDB), IBM Netezza®, Oracle and Teradata, in parallel and partitioned configurations
5. High performance and scalability for bulk and batch movement, and for real-time data replication and processing
6. Extensive tooling to support resource estimation, performance analysis and workload management
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 include:

- **Massive data scalability**: Organizations need to be confident that there are no limits when additional resources are added. If there is an upper boundary when resources are added, there will be business disruption and an inability to react and scale.

- **Linear (or better) performance improvements when adding hardware resources**: In other words, will additional resources lead to n-times better performance (where \( n = 2, 4, 8 \) and so on)?

- **Minimal non-hardware-related costs when the environment changes due to revised data characteristics or additional hardware resources**: To achieve the best return on investment for your data integration project, it is critical to avoid labor-intensive configuration changes when the environment changes. Adding processors or nodes to the hardware environment should occur with no change to the design of your data transformations, replication definitions and the end-to-end flow to avoid recompiling, retesting and deploying.

This paper presents essential elements for high-performance information integration and shows how they apply to business and technical decision makers responsible for designing, building, supporting and using scalable data processing systems.

**The case for parallelism**

To compete in today’s global business environment, organizations need a more granular level of detail (data)—and they need it faster to respond to changing market and competitive pressures. Consider these examples:

- 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 significant volumes of transactions per hour in each store; the data volume and performance requirements are enormous.

- Currency trading is a 24-hour business. Brokerage houses must 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 it is offering long-distance service from 2 to 14 in one year; the data volume and processing requirements for its marketing campaign’s data warehouse will greatly increase during that time.

**Information integration success stories**

Here is a small sampling of how IBM customers are using integration strategies and technologies to realize impressive benefits:

- A global banking customer processes 50,000 TB per second with complex transformations and guaranteed delivery

- Another global banking customer desensitizes 200 TB of data one weekend per month to populate development environments

- A third banking client shifted ETL-style data integration out of its data warehouse appliance to Hadoop using IBM InfoSphere Information Server, avoiding hand-coding for MapReduce and reducing development time by more than 80 percent

- A data services company powers its grid with InfoSphere Information Server, processing over 40 trillion records per month

- In healthcare, an information integration client uses text analytics across 200 million medical documents each weekend, creating indexes to support optimal retrieval by end users

- Another healthcare client built 200,000 programs in InfoSphere Information Server on a grid using low-commodity hardware

- An automaker coupled InfoSphere Information Server for data integration with a Hadoop-based landing zone, creating an authoritative source for more than 150 data marts

- A telecommunications company built a next-generation analytics infrastructure, including a central platform for all data integration activities, with governance extended to Hadoop
To support this growing data volume, variety and velocity, and the transitions from monthly or weekly batch runs to daily or up-to-the-minute requests, builders and consumers of enterprise data warehouses require a high-performance and scalable architecture. However, not all high-performance architectures are alike. Terms such as “parallel processing” and “scalability” carry different connotations and meanings from different vendors. 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 scalable, parallel information integration architecture are as follows:

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. An integrated platform for bulk and batch movement as well as real-time and trickle-feed processing
6. Extensive tooling for resource estimation, performance analysis and workload management
7. An extensible framework to incorporate in-house and third-party software

The rest of this paper refers to traditional information integration approaches as an architecture not supporting one or more of the key elements for a scalable, parallel architecture.

**Key element 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 online analytical processing (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 that allows developers to create visually a sequence of operations that can effectively manipulate data for quality and transformation purposes.

Data can come in from multiple data sources, such as flat files, databases, packaged applications (such as SAP, Salesforce, Oracle and so on) or as streams in real time. In all these cases, high throughput based on a data flow architecture remains important (see Figure 1).
As 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.

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 bottlenecks 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 and greatly increasing storage requirements
- 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.

**Key element 2: Dynamic data partitioning, in-flight data re-partitioning**

As described above, data pipelining is one approach to improve performance, particularly 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 its own limitations, particularly at the beginning and toward 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 complete.
Data partitioning is a complimentary approach to achieve parallelism, which distributes the load vertically among multiple instances of the data flow and against separate data partitions. The selected scope of source data is split into subsets called *partitions*. Multiple instances of end-to-end data flows that contain a sequence of operations then process the partition assigned to that instance.

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.

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

- Key (data) values
- Range
- Round-robin
- Random
- Entire
- Modulus
- Database matching partitioning (for example, 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. Traditional 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—for example, 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 the source data volume/characteristic changes. This can consume many weeks or months of development and testing prior to production. Plus, this labor-intensive effort has to be frequently repeated in dynamic environments.
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 (see Figure 6).

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 householding 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 on the fly based on the downstream process data partitioning, without landing the data to disk, which means this process is done in-memory—while data is also being pipelined to downstream processes when it is available.

Most traditional information integration tools cannot dynamically re-partition data; they require separate manual mappings for each process, which forces data to disk multiple times between steps 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 space, require more disk management and have greatly increased design complexity.

Key element 3: High scalability across a variety of hardware environments

Hardware vendors have offered scalable parallel computers for many years. Computing architectures span small, quad-core machines, multi-CPU systems, giant clusters and systems that have dedicated memory and disks. Here are some basic definitions.

**Symmetric multiprocessor (SMP)**

An SMP system is a tightly coupled multi-processor environment (a pool of homogeneous CPUs running independently) with a centralized shared memory.

**Clustered and massively parallel processor (MPP)**

Clustered environments and 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 a few to hundreds of processors.

**Grid computing**

With the commoditization of hardware computing power, grid computing is becoming a highly compelling option for large enterprises. In many organizations, traditional IT departments embark upon siloed information integration projects. Grid computing provides an outstanding alternative, allowing
Seven principles for achieving high performance and scalability for information integration

organizations to consolidate projects—bringing more processing power to bear and creating centers of efficiency and expertise—while reducing overall technology costs.

Grid computing takes advantage of all distributed computing resources—processor and memory—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 on the grid. What grid-computing software does best is balancing IT supply and demand by letting users specify their jobs’ CPU and memory requirements, and then finding machines on a network to meet those specs. Grid computing provides a set of horizontal integration capabilities that effectively addresses the challenge of cross-enterprise, cross-functional, IT resource integration and can extend a solution across multiple organizations or lines of business.

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 exponentially increases.

To make the best use of development resources, optimize use of hardware and avoid hitting performance walls down the road, IT organizations should insist 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 (the developer’s responsibility) and the mapping of that logic to the underlying parallel hardware platform (the data integration software’s responsibility).

Some data integration vendors using traditional information integration tools claim they can run on SMPs and MPPs. Although they may allow the deployment of data flows on different hardware configurations, they might require manual work to assign individual operations to run on a dedicated processor or to require manual redesign of the data flow into multiple “sub” data flows. This is not an approach that will lead to success—and there are some very important differences.

The first differentiating capability is the ability to design your data flow once (regardless of the number of processors on which it will be deployed), and then deploy flexibly. The same job should be able to be deployed without any changes to the job on a uniprocessor, SMP, MPP or grid environment. An efficient and effective information integration platform will streamline this process so that developers need only to change a configuration file to change the way the job is deployed, without even recompiling the data flows. Traditional information integration tools require labor-intensive changes and most likely a series of compile-test-tune deployment cycles. The end result with these tools is that any changes in data volume or system configuration will likely negatively impact the overall performance of the job.

Two other aspects to consider when considering an information integration solution include:

- Is the data integration platform able to truly saturate all the nodes of the MPP box (or the systems in the cluster/grid)?
- Can the information 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 four processors during the day and then in 20-way parallel mode at night, when additional resources are available, would this scenario actually require extensive rewriting of the data flow jobs? If information integration software does not seamlessly handle this, then the development team must do it—and maintain it—manually. In addition to driving up the cost of the project, this
also leads to an inability to maximize available hardware and spare computing power, as well as an inability to easily and flexibly scale as performance needs increase.

Most traditional information integration approaches cannot dynamically adapt to changes in the environment and automatically re-balance the load. Newly added hardware resources remain idle because to the user, a complex data flow cannot be automatically and transparently 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. When the information integration system does not automatically repartition the load to rebalance the overall workload, it will force the development team to:

- Recognize the change
- Understand how to change the design to better balance the load
- Make the change
- Test the modified design
- Deploy it
- Analyze the performance in the production environment
- Iterate through this change process multiple times until the workload is optimized, at least until the requirements change again

Obviously, this can be a time-consuming, disruptive and expensive undertaking. Even worse, many customers face dynamic environments where such changes occur frequently.

Without support for scalable hardware environments, the implications may involve:

- 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
- Disruption to the business

Key element 4: Parallel database connections

Enterprises with 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 (RDBMSs), such as IBM PureData™ (powered by IBM Netezza) or IBM 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 integration infrastructure should automatically support parallel access to leading databases such as DB2, Oracle, Netezza and TeraData. Productized database interfaces should support pulling and pushing multiple data streams in and out of the database—as well as running transaction logic—in parallel to avoid any sequential bottlenecks in processing. In addition, data partitioning should be consistent with how the database partitions the data (across nodes).
Seven principles for achieving high performance and scalability for information integration

Figure 7: Re-partitioning based on database partitioning.

Figure 7 shows data being re-partitioned before calling a *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.

Even files should be able to be 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 traditional information integration tools simply do not support parallel loading or 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 the complexities of the database.

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

- Extraction or loading will create a bottleneck of a single, sequential process, greatly slowing performance and minimizing the advantage of using a parallel database
- Data will be forced to disk to re-partition it before the load process, making the flow slower
- Developers will have to handle the complexities of the parallel database connections and re-partitioning

Depending on the information integration architecture supported, the application may be slower, require increased disk use and management, and greatly increase design complexity. These issues are magnified if developers are attempting to hand-code the information integration solution.

**Key element 5: Tooling for performance analysis and workload management**

The first important step to ensure high performance is to have sufficient resources for the required task. Insufficient resources (CPUs, disks and so on) have a significant impact on the overall performance. The architect and developer need to understand what the required resources will be, and identify potential bottlenecks ahead of time, before they deploy the data transformation process in a production environment. Resource estimation can increase performance significantly, and more importantly, can help organizations avoid multiple cycles of deploying a data flow on insufficient hardware, which adds more cost and complexity (that is, additional resources, redeployment, testing and so on). 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.
A second important step to ensure high performance is to manage workloads according to available resources and service-level agreements. An oversaturated environment (CPU or I/O) will impact the overall performance. Workload management ensures that a given workload is operating within its defined thresholds to enable optimal execution time and system utilization (see Figure 8).

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 (that is, transformation operations), what was the record throughput, what was the CPU utilization of individual stages, what was the memory utilization and so on. Graphical representation of 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 system problems and take the necessary actions to guarantee the highest levels of performance (see Figure 9).

**Key element 6: Additional integration requirements**

A scalable infrastructure should provide native, high-performance parallel components, particularly sorting, aggregation, joins, restructuring and so on. But because any large enterprise has special and customized needs, a scalable infrastructure should be extensible in order to integrate existing applications 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, Java and so on). A key requirement of integrating existing software code is the ability to operate only 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 runtime column (or schema) propagation—and it is a critical aspect to integrate existing applications without change, making them more portable and usable. With this ability, software can be integrated and parallelized.

Third-party tools and environments should also be able to be integrated and executed in parallel, such as Operational Decision Management, IBM InfoSphere® Optim™ Data Masking, master data management (MDM), IBM SPSS®, SAS or Apache Hadoop-based systems, to name a few. Many vendors claim to integrate with existing and third-party tools. They usually do this by landing data to disk, and then calling the “external” program. This is usually done by manually writing scripts, which is not an integrated solution and definitely is not executing them in parallel. A truly scalable architecture incorporates these non-native components and tools, taking advantage of data partitioning, dynamic re-partitioning and pipelining without landing data to disk between operations on any hardware environment.

Without an extensive framework that can incorporate additional flexible integration and data delivery styles, data:

- Cannot be integrated into the data flow
- Must be collected together back into one stream from its partitions and landed to disk
- Requires manual intervention to invoke the program sequentially
- Will force a restart of the next flow and new partitioning of the data

As a result, the application will be slower, require more disk capacity and management, and require manual intervention or script writing, thus greatly increasing 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. Without it, 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.

---

**Key element 7: Real-time replication and change data capture**

Data transformation has evolved 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 only the data that is changed. The change data is captured, transferred and transformed, and then applied to the target.
Four crucial real-time replication capabilities

- **High-speed bidirectional data**: Low-latency capture of real-time information
- **Noninvasive record capture**: Read data from transactional database logs and distribute data to any target—including big data streams, ETL for warehouses or IBM InfoSphere BigInsights™
- **Analyze a variety of information**: Analyze a variety of data in its native format—streaming audio, video and spatial, among others
- **Analyze extreme volumes of information in motion**: Terabytes per second and petabytes per day

Data replication’s real-time operational and analytical data synchronization can be used to enrich mobile applications and big data projects with the most current information, for example, or enable continuous availability across the data center or around the world. In heterogeneous environments, data replication supports data distribution and synchronization for transactional systems to support confident, up-to-the-minute decisions at the point of impact. Alternatively, when deployed in homogeneous environments, data replication can ensure business continuity and disaster recovery. In both scenarios, data replication minimizes the cost of infrastructure and optimizes resource utilization.

One important factor influencing performance and scalability in real-time data integration is the model to capture a change at a source. One option adopted by some technology providers is for the data integration 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 database must run queries (which may or not be indexed) to find changed records, rather than simply pushing the data already identified from the database log. The second and more efficient 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.

A range of options exists to capture a change in a source before pushing or publishing the change for further processing. These options range from simple trigger-based mechanisms to highly advanced log scraping technologies. The advantage of the log-based capture approach is a lower impact to the source database, which ultimately results in higher overall performance (see Figure 10). Instead of putting the burden of identifying the change on the database engine—for example, when using triggers—a dedicated, small-footprint CDC technology reads the changes directly from the database log file.

![Figure 10. An overview of low-impact data replication.](image-url)
The third important aspect of a CDC technology is whether or not data needs to be temporarily persisted between capturing the change and then processing it during data transformation and load into the target. One key capability to look for in a data replication solution is the ability to stream changes without persisting them along the way. This increases performance since the data does not need to be written to disk and then accessed from disk by a data transformation engine.

Finally, a CDC technology should be flexible:

- A CDC technology should speed the productivity of its users. It should provide graphical configuration for ease of use and scripting for automated configuration (see Figure 11).

- A CDC technology should offer a variety of integration methods depending on volume and performance requirements. These may include file-based, queue-based, through a staging table or direct. It should be able to be run on the source or remotely for minimum impact.

- A CDC technology should take into consideration the targets it is applying data to and adjust accordingly. For instance, some technology providers use a one-size fits-all approach to applying data, leaving it to the users to tune from there.

- A CDC technology should also offer the ability to apply data in parallel in order to boost performance, taking into consideration all the previously mentioned benefits of parallelism.

- A CDC technology with broad heterogeneous support will ensure that the built-in performance and scalability features can be leveraged across the enterprise.
Case in point: Soitec Solar Industries, LLC improves quality and throughput
Soitec Solar Industries, LLC, an international manufacturer, produces high-performance materials for the semiconductor industry. The company’s market-leading products and technologies are designed to enhance the performance and energy efficiency of microelectronics-driven IT, communications, automotive electronics, lighting products and solar power plants for large-scale utilities. Soitec has manufacturing plants and research and development (R&D) centers in France, Singapore, Germany and the United States.

Soitec is building a new plant to manufacture its Concentrator Photovoltaic (CPV) technology for a new solar power plant in Southern California. The company needed to balance a need for high throughput in its manufacturing processes with the imperative for quality control of the CPV module components. It sought a turnkey manufacturing execution system to track and monitor the performance of its manufacturing equipment that would also detect and intelligently respond to even small variances in product quality.

The process for manufacturing solar cells for use in power plants needs to be exact, because even slight variations in the quality of the finished product can reduce its efficacy. This solar module manufacturer implemented an automated manufacturing execution system (MES) that uses near–real-time data from equipment sensors to monitor equipment performance and product quality. The solution analyzes the data to detect trends or shifts in quality based on product and process parameters.

If detected, a smart tool control system can take automatic, intelligent corrective actions like stopping equipment or even the complete production line to prevent yield loss and allow retooling based on precise information from the MES system. Complete production visibility by the MES helps ensure both higher quality and more efficient production for the company.

Soitec has realized the following benefits:

- Generates 400 percent of the production throughput of similar factories without automated MES systems
- Detects quality variances early in the manufacturing process nearly 100 percent of the time, replacing postproduction quality inspections that yield far fewer results
- Enables the traceability and proactive recall of any modules that develop quality defects later during solar plant operations

Summary
Big data streams in at high velocity—so performance and scalability are critically important. Data changes rapidly, and it must be fed to various applications quickly so business leaders can react to changing market conditions as soon as possible. To successfully handle the ever-increasing volumes and complexity of data, you need an enterprise-class data integration solution that can meet stringent and fluctuating requirements in a flexible and cost-efficient way.

This paper introduced seven elements that are critical to ensure the highest degree of performance and scalability for your information integration deployment. To ensure confidence, your information integration solution should be able to meet all seven elements.

Each of the key capabilities highlighted in the paper are supported as part of the IBM information integration platform, which includes InfoSphere Information Server, InfoSphere Federation Server and InfoSphere Data Replication.
For more information
To explore how IBM can help your organization realize the promise of information integration for complex or high-performance projects, including grid deployments, contact your IBM representative or IBM Business Partner, or visit:

- IBM Information Integration:
  ibm.com/software/data/integration
- IBM Data Integration:
  ibm.com/software/products/en/category/SWB50
- IBM InfoSphere Information Server for Data Integration:
  ibm.com/software/products/en/infoservfordatainte
- IBM Grid Computing:
  ibm.com/grid
- IBM BladeCenter®:
  ibm.com/systems/bladecenter
- InfoSphere Data Replication:
  ibm.com/software/data/integration

Additionally, IBM Global Financing can help you acquire the software capabilities that your business needs in the most cost-effective and strategic way possible. We'll partner with credit-qualified clients to customize a financing solution to suit your business and development goals, enable effective cash management, and improve your total cost of ownership. Fund your critical IT investment and propel your business forward with IBM Global Financing. For more information, visit:
ibm.com/financing

© Copyright IBM Corporation 2014
IBM Corporation
Software Group
Route 100
Somers, NY 10589
Produced in the United States of America
April 2014
IBM, the IBM logo, ibm.com, BigInsights, BladeCenter, DB2, InfoSphere, Optim, PureData, and SPSS are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at ibm.com/legal/copytrade.shtml

Netezza® is a trademark or registered trademark of IBM International Group B.V., an IBM Company.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

THE INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT. IBM products are warranted according to the terms and conditions of the agreements under which they are provided.

Statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Please Recycle