



# The Netezza FAST Engines™ Framework

A Powerful Framework for High-Performance Analytics



## Introduction

Companies around the world who run their businesses on Netezza Performance Server® (NPS®) streaming analytic™ appliances rely on Netezza's industry-leading analytic price/performance and simplicity for their real-time and complex analytic and data warehousing needs. Through its unique Asymmetric Massively Parallel Processor™ (AMPP™) architecture, the NPS system combines the operational simplicity of an SMP node for administration with the raw performance horsepower of an MPP grid of intelligent storage nodes, each containing its own disk drive, CPU, memory and a key programmable hardware element known as a Field Programmable Gate Array (FPGA). As a result, the system executes analytical queries at streaming speeds – as rapidly as data can be read from disk – by leveraging the FPGA technology as a critical system performance multiplier.

The NPS performance multiplier effect is the result of a framework of FPGA-Accelerated Streaming Technology (FAST) Engines™ that leverage the embedded FPGA to provide performance acceleration advantages and functionality as rapidly as data can be read (or "streamed") from the disk drive on each intelligent node.

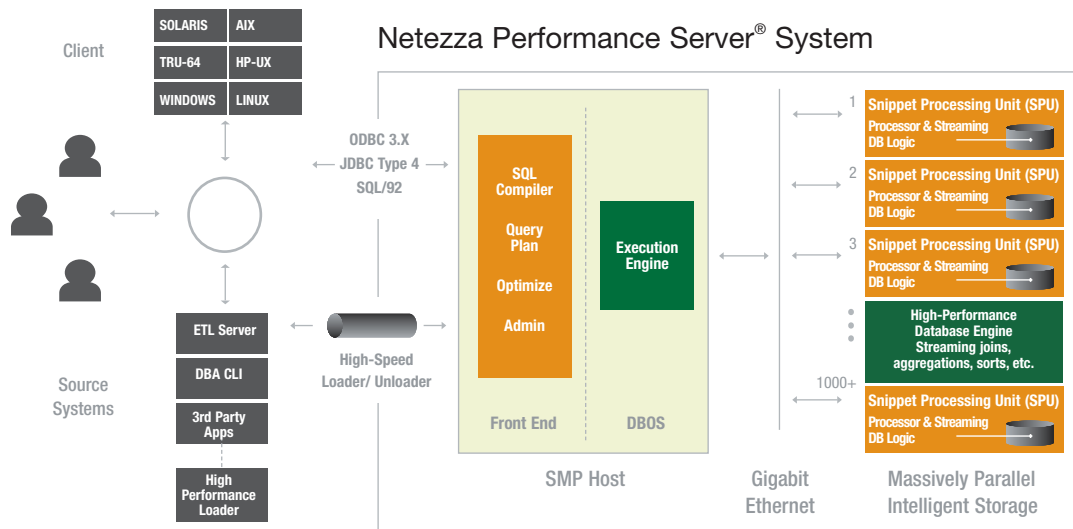
FAST Engines already provide significant price/performance and scalability differentiation for the NPS family of appliances. Like applying a turbo-charger to an already powerful engine, Netezza is now introducing its newest FAST Engine, Compress, which uses patent-pending technology to deliver yet another 100-200% performance increase in streaming analytic performance. Netezza's Compress Engine is designed primarily for performance improvement. Rather than the CPU-intensive compression efforts employed by other vendors to reduce disk usage that also result in reduced performance, the Compress Engine *accelerates* performance.



As an extensible framework within the NPS appliance, FAST Engines lay the foundation for ongoing innovation, new product capabilities and further performance enhancement. Through an expanding set of FAST Engines, Netezza is creating a broadened role for its streaming analytic appliances in terms of the data sizes, data types and analytic challenges that can be brought "into the appliance".

## The Netezza Architecture - Designed for Streaming Speeds

Netezza's AMPP™ (Asymmetric Massively Parallel Processing) system architecture, the best combination of MPP and SMP, provides orders of magnitude performance improvements without the complexity, tuning, indexing and aggregations necessary in other competing data warehouse and analytic systems.



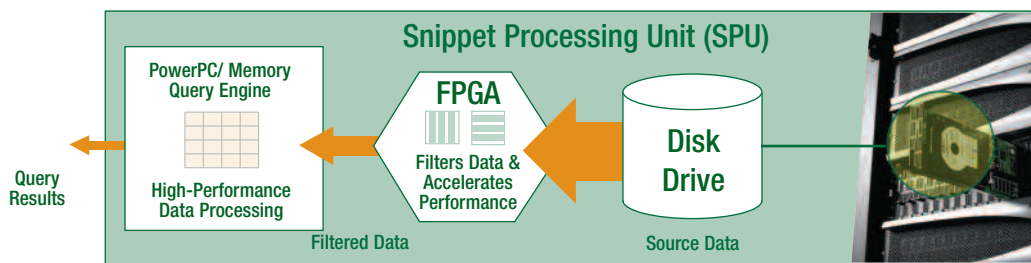
A major part of the performance advantage derives from the system's MPP architecture – today allowing up to nearly 900 intelligent storage nodes to "divide and conquer" the workload and provide responses to a broad range of queries, from simple, tactical queries to operational queries running in near real-time to deep analytics.

A second, and even more critical, performance advantage lies in the architecture of the intelligent storage nodes themselves and how the NPS system software makes use of them. In the Netezza appliance, these nodes are known as "Snippet Processing Units" (or "SPUs") – each with its own embedded disk drive, CPU, memory and also a common off-the-shelf device known as a Field Programmable Gate Array (FPGA).

The balanced approach of the SPU enables each node to perform streaming analytic processing – even with an extremely busy workload. Each SPU is capable of handling many concurrent query snippets from multiple queries, simultaneously streaming from disk, processing in the CPU and memory and/or moving results across the internal backbone network of the NPS system.

Combined with the system's sophisticated software resource optimization, the hardware architecture of the NPS appliance provides industry-leading analytic query performance. The focus of the system's optimization is to enable streaming processing; in effect, processing and retiring analytic operations as rapidly as the relevant information for them can be read from the many parallel disk drives in the system.

The FPGA is central to the FAST Engines framework performing critical filtering and query processing function, *as fast as data streams from the disk drive on each SPU*. The typical impact of this work is a reduction of 95% or more in the data required for further processing by the on-board CPU and memory.

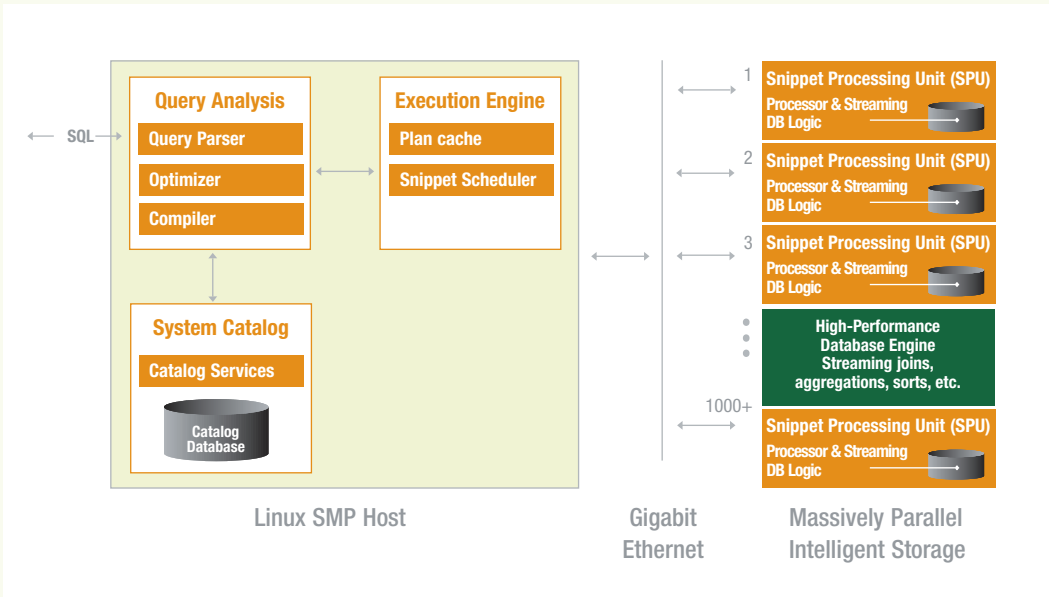


**Netezza Intelligent Query Streaming**  
Architecturally reducing data movement & delivering high performance

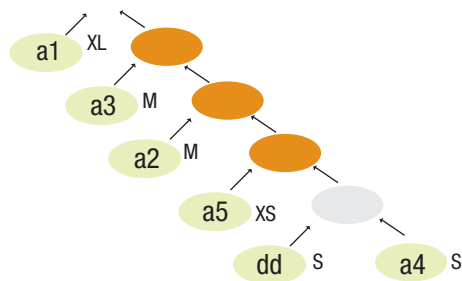
As a commodity technology component, the FPGA is found in just about any "streaming" data product in the market today: from digital video recorders and DVD players to automotive electronics and displays to telecommunication switches to high-performance computing systems. As its name suggests, this small, low-power device is a highly reconfigurable, extensible functional element in the design of those products. Because of its widespread use the FPGA also enjoys a very robust technology curve, with projected five-year rates of price/performance enhancements that exceed those predicted for CPU technology by Moore's Law.

## Query Processing in the NPS System - a Primer

How does query processing work inside an NPS appliance? As SQL queries arrive at an SMP host in the appliance, an optimized query plan is created in the host based on a cost-based optimization algorithm that is designed to understand the unique capabilities of the NPS system architecture.



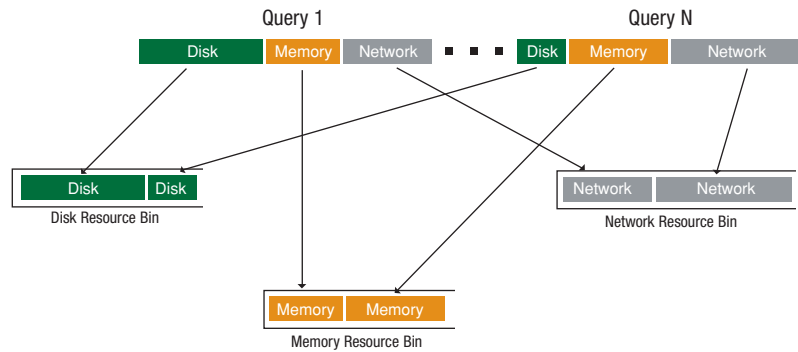
Wherever possible, the optimizer moves processing to the MPP grid of SPUs to leverage maximum system performance while also attempting to limit the amount of disk read activity, memory use and data movement required within the system to complete the task. In so doing, the optimizer chooses the join order and query streaming for the query execution plan. This plan is composed of a sequence of smaller, atomic-level database functions (e.g., scan, join, sort) that will result in the completion of the overall query processing.



A "Bubble" View of a Five-snippet, Six-table Query Plan

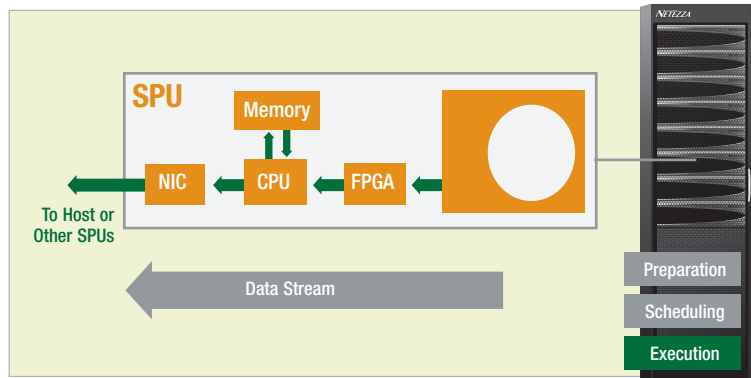
The snippets are then compiled as C++ code or extracted from a large cache of previously compiled, *'parameterizable'* snippet object files. Because compilation is done at the snippet level and support parameter variations (e.g., a snippet using the clause, "where name='bob'" might use the same compiled code as a snippet using the clause, "where name='jim'" but with differing parameter settings), this use of a snippet cache eliminates the compilation step for over 99% of the snippets in an operating system, greatly accelerating system performance.

For each snippet, there are two elements that are distributed to the SPUs: the compiled snippet file and a set of FPGA parameters that will customize the FAST Engines for maximum efficiency on that particular snippet. These are scheduled by the system to maximize throughput based on system load, response time and workload management settings and then broadcast to the SPUs for parallel execution.



"Bin-Packing" SPU Scheduling of Multiple Concurrent Snippet Processes

The SPUs then execute the snippet as required, using built-in functionality to speed performance, including ZoneMap™ acceleration to limit the amount of data read, and the FAST Engines in the FPGA to eliminate the unnecessary records and columns. The data that remains to be processed to complete the snippet step and move on to the next snippet is typically an extremely small fraction of the full table data from which it came.



Stream Processing and Data Reduction on Each SPU

The process of scheduling and executing the snippets then continues through to the completion and retirement of the query, with results returned through the host to the application layer.

## The FAST Engines Framework

The FPGA is a critical enabler of the price/performance (and space and power efficiency) advantages of the NPS system, and is at the heart of Netezza's patented streaming architecture. By using the FPGA to handle important functions and filtering away unneeded data as it's read from the disk drive on each SPU, the system can deliver much higher performance in a more space- and power-efficient footprint by putting less computational stress on the CPU and memory of each SPU. Those primary functions that are built into the FPGA in the NPS system are referred to as "engines" and compose the FAST Engines framework.

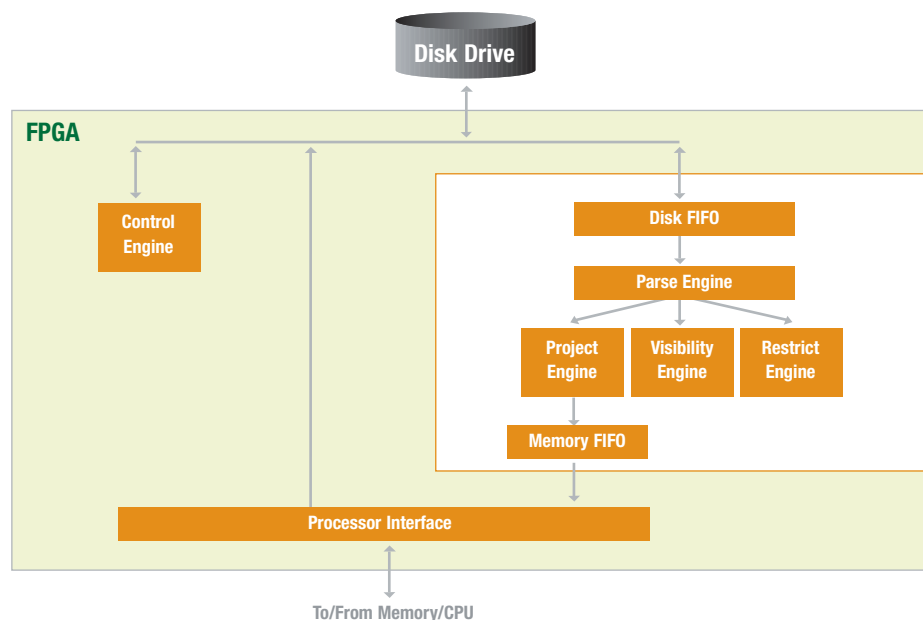
The FPGA-Accelerated Streaming Technology (FAST) Engines Framework is based on three main concepts:

- FAST Engines are basic analytic functions electronically programmed into the FPGA to accelerate query performance;
- FAST Engines are dynamically reconfigurable; and
- FAST Engines are customized at run-time for each snippet executed in the SPU.

*Dynamic reconfigurability* allows each of the engines embedded in the FPGA to be modified, disabled or extended by the NPS system.

*Run-time customization* enables the FPGA to incorporate parameters passed to each engine to optimize the behavior of the FPGA for a particular query snippet. By enabling this snippet-by-snippet customization, one could think of the NPS platform as providing an "*optimized hardware configuration for each snippet*".

The five current FAST Engines in the framework include Control, Parse, Visibility, Project and Restrict. These engines work in a combined serial and parallel manner, as seen in the following logical block diagram and their overall performance multiplier effect drives the NPS system.



**Control Engine:** The "Control" Engine controls the hard disk and manages direct memory access (DMA) data flows from the disk to the on-board SPU memory. It manages reads and writes to the disk drive and data flows to memory, under on-SPU software direction. Storage management performance enhancement functions, such as ZoneMap acceleration to reduce the number of data blocks read from the disk, are enforced through the Control Engine.

**Parse Engine:** The "Parse" Engine encompasses two key roles in the FAST Engines framework. First, it provides Error-Correcting Code (ECC) checking and correction as data streams off the disk drive, providing a more robust disk interface and reduced need to retry disk reads. Second, it parses the incoming data stream from the disk drive and passes the appropriate data to each of the remaining three downstream engines: Visibility, Project and Restrict.

**Visibility Engine:** The "Visibility" Engine provides the ACID (Atomicity, Consistency, Isolation and Durability) isolation enforcement for the NPS appliance. As data are streamed from the disk drive, the Visibility Engine will eliminate records (rows) of data that should not be visible to the particular query snippet being executed - either because the records had been marked deleted by an earlier query or because they had been added to the database after the start of the current query.

What this means is that ACID isolation is fully automated and customized on a per-snippet basis, at data streaming rates and without expending any additional CPU or memory cycles. Data that should not be visible for processing is immediately filtered away and eliminated in this FPGA engine.

**Project Engine:** The "Project" Engine enforces filtering of the column data read from the disk based on the "SELECT" clause of the SQL statement. Only columns included by the clause are allowed to continue for further processing and any columns not selected will be filtered and eliminated prior to any records being sent to the on-board CPU and memory.

Here, particularly for wide base tables containing many columns, there is significant query acceleration through the filtering of unnecessary data. Rather than requiring excessive uses of the memory or CPUs on each SPU, the FPGA filters away the data as fast as it streams from the disk.

**Restrict Engine:** The "Restrict" Engine enforces filtering of record (row) level data read from the disk based on the "WHERE" predicate clauses of the SQL statement. Only records that could satisfy the clauses will be allowed to continue for further processing and any records not selected are filtered and eliminated prior to being sent to the on-board CPU and memory.

The Restrict Engine typically results in an exceptionally high degree of data record filtering, greatly reducing the follow-on snippet processing work required of the CPU and memory on each SPU.

## FAST Engines Performance - An Example

The performance multiplying effects of these Engines is clearly shown through a short example. Consider the following simple SQL query:

```
Select state, gender, age, count(*) From 8_billion_Row_Table
Where dob < '04/01/2000' And dob > '12/31/1999' And zip =
32605 Group by state, gender, age;
```

In this example, first the storage manager and FPGA would use ZoneMap acceleration to limit the disk read to only those disk extents on each SPU with dates of birth occurring in the three-month period of January through March 2000, rather than scanning the full table.

Then, when the data was read from the disk, the FPGA would apply the *Visibility* Engine to restrict the records being investigated to only those that met the ACID isolation requirements.

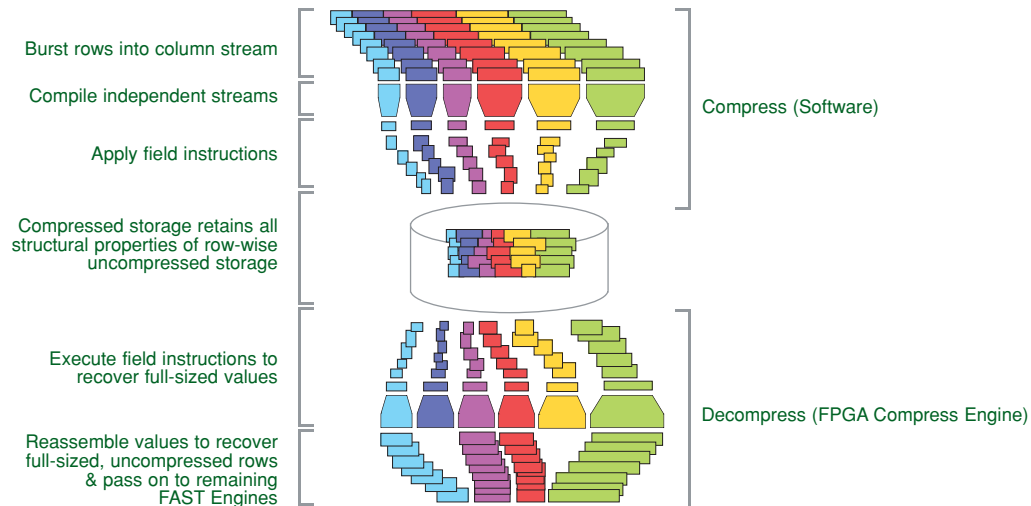
It would further *restrict* the rows of data returned to those records within the three-month range and a zip code matching the query and finally, the column data *projected* to the memory and CPU would be limited to only state, gender and age information of each record.

If the table in question contained 100 or more columns for each record, the *Project* Engine results could represent less than 3% of the column data. If one assumes the table in question contained birthdate information for just the last seven years, the *Restrict* Engine would dramatically reduce the row-count of data delivered to memory/CPU as well - specifically by more than 25:1, or 3 months out 84.

**The net FAST Engines performance impact would be that the on-board CPU and memory would be responsible for further snippet processing of just 0.12% (or twelve record elements per 10,000) of the data in the base table.**

## “Compress”- the Next FAST Engine

The next engine to be introduced into the FAST Engines framework will be the "**Compress**" Engine.

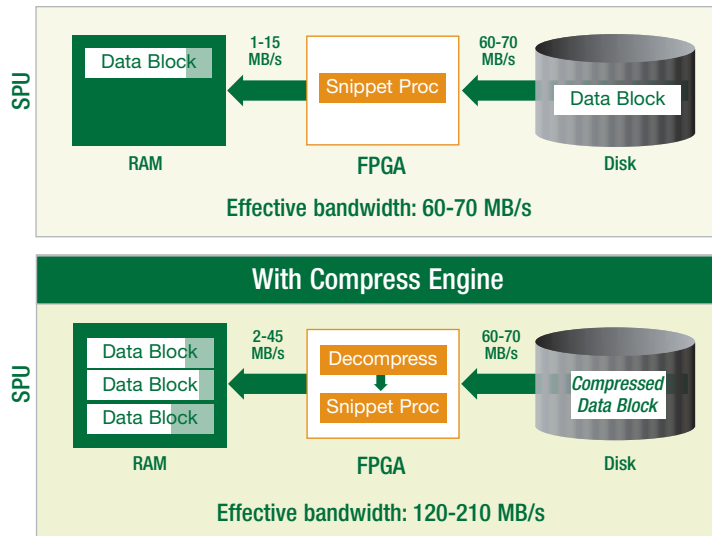


The Compress Engine will have the net effect of boosting streaming query performance by 2-3x (100-200%) through a software enhancement to the NPS appliance. Employing a patent-pending method for compiling columnar data in all tables of the database, the NPS system will be able to compress the data to use disk much more effectively and greatly increase query performance as data streams from the disk.

Unlike traditional compression approaches used by several of the competitive data warehousing vendors, Netezza's Compress Engine is designed primarily for performance improvement, and not merely to reduce data sizes and system footprints. In fact, other compute-intensive approaches typically mitigate any performance gains, or result in a performance degradation to accomplish the compression effects, whereas Netezza's approach delivers a significant boost in performance.

The Compress Engine, like all other FAST Engines, is dynamically configurable and optimized at run-time. As data is written to disk (e.g., during data load, insert or update operations) it is compressed into a compiled format, column-by-column with the original data replaced by the Compress Engine "instruction set" for decompilation.

As this data is read from the disk, the Compress Engine reads its instruction set and reassembles the original data as it streams from the disk, effectively raising the streaming data rate by as much as 200% - lifting the effective scanning rate per SPU node from over 60 MB/sec to approximately 200 MB/sec.



The net effect of this FAST Engine enhancement will be to lift persistent data scan rates to as much as 70 TB/hour per NPS system rack, or approximately 560 TB/hour in the 8-rack 10800 system configuration – without requiring any change in NPS hardware.

Another benefit of the Compress Engine is that it is extensible. Using the same basic compilation algorithm, the Compress Engine could be throttled up to provide an even higher level of data streaming performance gain and/or could also be used to provide additional functionality such as support for at-rest data Encryption.

## A Framework for Ongoing Innovation

The FAST Engines framework also provides a foundation for extending the NPS system's price/performance and functionality even further in the future. Through the development of additional Engine innovations, Netezza can provide customers with powerful capabilities that can be dynamically configured in the FPGA via a straightforward software upgrade. Just a few of the possibilities for future FAST Engines include:

- A Multi-Level Security Engine to enforce row- & column- level security/access. This could be thought of, in basic terms, as an extension of Visibility Engine discussed previously.
- A Join Filtering Engine to perform primary join row elimination. This new Engine would filter rows that do not meet single or complex multiple join conditions as they are streamed from disk – boosting performance even further.
- An extension to the new Compress Engine to provide as much as another 100% improvement in query streaming performance.
- An Encrypt Engine to support on-stream encryption/decryption of data at rest. This engine could be realized as an extension of the patent-pending compilation algorithms used in the new Compress Engine.
- Additional Non-SQL processing engines that will arise from the Netezza Developer Network activities. Already one partner is using the current FPGA development toolset to provide streaming image analysis and pattern-match scoring based on its own proprietary FPGA engine development. With new progress into the research and developments toward direct C-to-FPGA programmability, more advanced FPGA based non-SQL engines will become possible to a broader set of applications developers and customers.

## Summary

The NPS streaming analytic appliance relies on Netezza's AMPP architecture to deliver industry-leading price performance. In turn, the AMPP architecture is powered by the FAST Engines framework. FAST Engines have provided a critical performance multiplier in the NPS system since the first generation of Netezza appliances, and Netezza is now adding a major new FAST Engine to the platform that will boost performance as much as 100-200%. The new Compress Engine is a fundamentally different approach from traditional approaches to data compression, in that it offers a pure performance gain for query processing in addition to the benefits of a more efficient use of disk. The FAST Engines framework also provides the basis for extending the system's industry-leading capabilities even further in the future, making possible additional innovations in system performance, new product capabilities and the embedding of advanced analytics functionality to power Netezza customers to a new level of business analytics and real-time decision-making.

### About Netezza

Netezza (NYSE Arca: NZ) is the global leader in analytic appliances that dramatically simplify high-performance analytics for business users across the extended enterprise, delivering significant competitive and operational advantage in today's information-intensive marketplaces. The Netezza Performance Server® (NPS®) family of streaming analytic™ appliances brings appliance simplicity to a broad range of complex data warehouse and analytic challenges. Customers who are realizing the benefits of Netezza appliances include Ahold, Amazon.com, CNET Networks, Debenhams, Department of Veterans Affairs, Epsilon, Neiman Marcus, Orange UK, Premier, Inc., Ross Stores, Ryder System, Inc., The Carphone Warehouse and Virgin Media. Based in Framingham, Mass., Netezza has offices in Washington, DC, the United Kingdom and Asia Pacific.

**For more information about Netezza, please visit [www.netezza.com](http://www.netezza.com).**



Netezza Corporation : 200 Crossing Boulevard : Framingham, MA : 01702-4480  
+1 508 665 6800 tel : +1 508 665 6811 fax : [www.netezza.com](http://www.netezza.com)