RC25317 (WAT1209-067) September 22, 2012
Mathematics

IBM Research Report

Analyzing Analytics

Rajesh Bordawekar[1], Bob Blainey[2], Chidanand Apte[1]

[1]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 208
Yorktown Heights, NY 10598
USA

[2]IBM Toronto Software Lab
8200 Warden Avenue
Markham, Ontario L6G 1C7
Canada

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Analyzing Analytics

Rajesh Bordawekar
IBM Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532
bordaw@us.ibm.com

Bob Blainey
IBM Toronto Software Lab
8200 Warden Avenue
Markham, Ontario L6G 1C7
blainey@ca.ibm.com

Chidanand Apte
IBM Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
apte@us.ibm.com

1 Math is cool again

From streaming news updates on smart-phones, to instant messages on micro-blogging sites, to posts on social network sites, we are all being overwhelmed by massive amounts of data [33, 27]. Access to such a large amount of diverse data can be of tremendous value if useful *information* can be extracted and applied rapidly and accurately to a problem at hand. For instance, we could contact all of our *nearby* friends for a dinner at a local *mutually agreeable* and *well-reviewed* restaurant that has advertised *discounts* and *table availability* for *that night*; but finding and organizing all that information in a short period of time is very challenging. Similar opportunities exist for businesses and governments but the volume, variety and velocity of data can be far greater. This process of identifying, extracting, processing, and integrating *information* from *raw data*, and then applying it to solve a problem is broadly referred to as *analytics*.

Table 1 presents a sample of analytic applications from different domains, along with their functional characteristics. As this Table illustrates, many services that we take for granted and use extensively in everyday life would not be possible without analytics. For example, social networking applications such as Facebook, Twitter, and LinkedIn encode social relationships as graphs and use graph algorithms to identify hidden patterns (e.g. finding common friends). Other popular applications like Google Maps, Yelp or FourSquare combine location and social relationship information to answer complex spatial queries (e.g. find the nearest restaurant of a particular cuisine that your friends like). Usage of analytics has substantially improved the capabilities and performance of gaming systems as demonstrated by the recent win of IBM's Watson/DeepQA intelligent question-answer system over human participants in the Jeopardy challenge [29]. The declining cost of computing and storage and the availability of such infrastructure in *cloud* environments has enabled organizations of any size to deploy advanced analytics and to package those analytic applications for broad usage by consumers.

While consumer analytical solutions may help us all to better organize or enrich our personal lives, the analytic process is also becoming a critical capability and competitive differentiator for modern businesses, governments and other organizations. In the current environment, organizations need to make on-time, informed decisions to succeed. Given the globalized economy, many businesses have supply chains and customers that span multiple continents. In the public sector, citizens are demanding more access to services and information than ever before. Huge improvements in communication infrastructure have resulted in wide-spread use of online commerce and a boom in smart, connected mobile devices. More and more organizations are run around the clock, across multiple geographies and time zones and those organizations are being *instrumented* to an unprecedented degree. This has resulted in a deluge of data that can be studied to harvest valuable information and make better decisions. In many cases, these large volumes of data must be processed rapidly in order to make timely decisions. Consequently, many organizations have employed analytics to help them decide what kind of data they should collect, how this data should be analyzed to glean key information, and how this information should be used for achieving their organizational goals. Examples of such techniques can be found in almost any sector of the economy, including financial services [7, 6], government [31, 13], healthcare, retail [26, 23], manufacturing, logistics [1, 19], hospitality, and eCommerce [8, 9].

The distinguishing feature of an analytics application is the use of mathematical formulations for modeling and processing the raw data, and for applying the extracted information [32]. These techniques include statistical approaches, numerical linear algebraic methods, graph algorithms, relational operators, and string algorithms. In practice, an analytics application uses multiple formulations, each with unique functional and runtime characteristics (Table 1). Further, depending on the functional and runtime constraints, the same application can use different algorithms. While many of the applications process a large volume of data, the type of data processed varies considerably. Internet search engines process unstructured text documents as input, while retail analytics operate on structured data stored in relational databases. Some applications such as Google Maps, Yelp, or Netflix use both structured and unstructured data. The velocity of data also differs substantially across analytics applications. Search engines process read-only historical data whereas retail analytics process both historical and transactional data. Other applications, such as the monitoring of medical instruments, work exclusively on real-time or streaming data. Depending on the mathematical formulation, the volume and velocity of data and the expected I/O access patterns, the data structures and algorithms used by analytical applications vary considerably. These data structures include vectors, matrices, graphs, trees, relational tables, lists, hash-based structures, and binary objects. They can be further tuned to support in-memory, out-of-core, or streaming execution of the associated algorithm. Thus, analytics applications are characterized by diverse requirements but share a common focus on the application of advanced mathematical modelling, typically on large data sets.

Although analytics applications have come of age, they have not yet received significant attention from the computer architecture community. It is important to understand systems implications of the analytics applications, not only because of their diverse and de-

Application	Domain	Principal Goals	Key Functional Characteristics
Netflix and Pandora [3, 18]	Consumer	Video and music recommendation	Analyzing structured and unstructured data, Personalized recommendations
Yelp, FourSquare	Consumer	Integrated geographical analytics	Spatial queries/ranking, Streaming and persistent data
DeepQA (Watson) [11]	Healthcare Consumer	Intelligent question-answer (QA) System	Real-time natural language, Unstructured data processing, Artificial intelligence techniques for result ranking
Telecom Churn Analysis [26]	Telecom	Analysis of call-data records	Graph modeling of call records, Large graph dataset, Connected component identification
Fraud Analytics	Insurance Healthcare	Detection of suspicious behavior	Identification of abnormal patterns, Real-time data analysis over streaming and persistent data
Cognos Consumer Insight [28] Twitter Sentiment [12]	Marketing Hospitality	Sentiment/Trend Analysis of BLOGS and text messages	Processing large corpus of text documents, Extraction and transformation, Text indexing, Entity extraction
UPS [1], Airline Scheduling [19]	Logistics	Transportation routing	Mathematical programming solutions for transportation
Salesforce.com	Marketing	Customer data analytics	Reporting, Text search, Multi-tenant support, automated price determination, recommendation
Moody's, Fitch, S&P [7, 6]	Financial	Financial credit rating	Statistical analysis of large historical data
Oracle, Amazon Retail Analysis	Retail	End-to-end retail management	Analysis over large persistent and transactional data, Integration with logistics and customer Information
Energy Trading	Energy	Determining and hedging prices	Processing large time-series data, Integrated stochastic models for generation, storage and transmission
Splunk [30]	Enterprise	System management analysis	Text analysis of system logs, Large data sets
Flickr, Twitter, Facebook and Linkedin	Consumer Enterprise	Social network analysis	Graph modeling of relations, Massive graph datasets, Graph analytics, Multi-media annotations and indexing
Voice of Customer Analytics [4]	Enterprise	Analyzing customer voice records	Natural language processing, Text entity extraction
Facial Recognition [31]	Government	Biometric classification	Analysis and matching of 2-/3-D images, Large data sets
Predictive Policing [20]	Government	Crime prediction	Spatial and temporal analytics

Table 1. Well-known analytics solutions and their key characteristics

manding requirements, but also, because systems architecture is currently undergoing a series of disruptive changes. Wide-spread use of technologies such as multi-core processors, specialized co-processors or accelerators, flash memory-based solid state drives (SSDs), and high speed networks has created new optimization opportunities. More advanced technologies such as phase-change memory are on the horizon and could be game-changers in the way data is stored and analyzed. In spite of these trends, currently there is limited usage of such technologies in the analytics domain. Even in the current implementations, it is often difficult for analytics solution developers to fine-tune system parameters, both in hardware and software, to address specific performance problems. Naive usage of modern technologies often leads to unbalanced solutions that further increase optimization complexity.

Thus, to ensure effective utilization of system resources: CPU, memory, networking, and storage, it is necessary to evaluate analytics workloads in a holistic manner. We aim to understand the application of modern systems technologies to optimizing analytics workloads by exploring the interplay between overall system design, core algorithms, software (e.g., compilers, operating system), and hardware (e.g., networking, storage, and processors). Specifically, we are interested in isolating repeated patterns in analytical applications, algorithms, data structures, and data types, and using them to make informed decisions on systems design. Over the past two years, we have been examining the functional flow of a variety of analytical workloads across multiple domains (Table 1), and as a result of this exercise, we have identified a set of commonly-used analytical models, called *analytics exemplars* [5]. We believe that these exemplars represent the essence of analytical workloads and can be used as a toolkit for performing exploratory systems design for the analytics domain. We use these exemplars to illustrate that analytics applications benefit greatly from holistically co-designed software and hardware solutions and demonstrate this approach using the Netezza [16] appliance as an example. Finally, we hope this study acts as a *call to action* for computer architects and systems designers to focus future research on analytics.

2 Anatomy of Analytics Workloads

To motivate the study of analytics workloads, we first describe in detail a recent noteworthy analytics application: the Watson intelligent question/answer (Q/A) system [11]. Watson is a computer system developed to play the Jeopardy! game-show against human participants [29]. Waston's goals are to correctly interpret the input natural language questions, accurately predict answers to the input questions and finally, intelligently choose the input topics and the wager amounts to maximize the gains. Watson is designed as an open-domain Q/A system using the DeepQA system, a probabilistic evidence-based software architecture whose core computational principle is to assume and pursue multiple interpretations of the input question, to generate many plausible answers or hypotheses and to collect and evaluate many different competing evidence paths that might support or refute those hypotheses through a broad search of large volumes of content. This process is accomplished using multiple stages: the first, question analysis and decomposition stage parses the input question and analyzes it to detect any semantic entities like names or dates. The analysis also identifies any relations in the question using pattern-based or statistical approaches. Next, using this information, a keyword-based primary search is performed over a varied set of sources, such as natural language documents, relational databases and knowledge bases, and a set of supporting passages (initial evidence) is identified. This is followed by the candidate (hypothesis) generation phase which uses rule-based heuristics to select a set of candidates that are likely to be the answers to the input question. The next step, Hypothesis and Evidence Scoring, for each evidence-hypothesis pair, applies different algorithms that dissect and analyze the evidence along different *dimensions of evidence* such as time, geography, popularity, passage support, and source reliability. The end result of this stage is a ranked list of candidate answers, each with a confidence score indicating the degree to which the answer is believed correct, along with

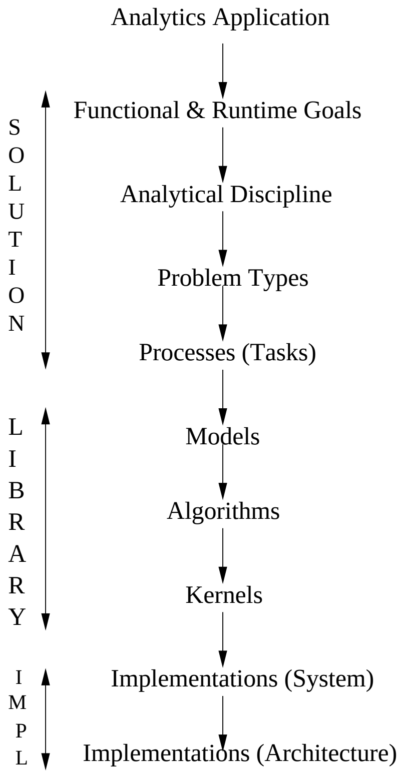

Figure 1. Simplified functional flow of business analytics applications

links back to the evidence. Finally, these evidence features are combined and weighted by a logistic regression to produce the final confidence score that determines the successful candidate (i.e. the correct answer). In addition to finding correct answers, Watson needs to master the strategies to select the clues to it's advantage and bet the appropriate amount for any given situation. The DeepQA system models different scenarios of the Jeopardy! game using different simulation approaches (e.g., Monte Carlo techniques) and uses the acquired insights to maximize Watson's winning chances by guiding topic selection, answering decisions and wager selections.

The Watson system displays many traits that are common across analytics applications. They all have one or more functional goals. These goals are accomplished by one or more multi-stage processes, where each stage is an independent analytical component. To study the complex interactions between these components, it is useful to examine the functional flow of an analytics application from the customer usage to implementation stages. As Figure 1 illustrates, execution of an analytics application can be partitioned into three main phases: (1) solution, (2) library, and (3) implementation. The solution phase is end-user focused and customized to to satisfy user's functional goals, which can be one of the following: prediction, prescription, reporting, recommendation, quantitative analysis, simulation, pattern matching, or alerting[1]. For example, Watson's key functional goals are: *pattern matching* for input question analysis, *prediction* for choosing answers, and *simulation* for wager and clue selection. Usually, any functional goal needs to be achieved under certain runtime constraints, e.g., calculations to be completed within a fixed time period, processing very large datasets or large volumes of data over streams, supporting batch or ad-hoc queries, or supporting a large number of concurrent users. For example, for a given clue, the Watson system is expected to find an answer before any of the human participants in the quiz. To achieve the functional and runtime goals of an application, the analytical solution leverages well-known analytical disciplines such as machine learning, data mining, statistics, business intelligence, and numerical analysis. Specifically, for a given analytical problem, the solution chooses appropriate problem types from these disciplines to build processes. Examples of analytic problem types include supervised and unsupervised learning, optimization, structured and unstructured data analysis, inferential and descriptive statistics, and modeling and simulation.

Table 2 presents a set of analytics applications along with their functional goals and the analytic problem types used to achieve these goals. As illustrated in Table 2, in many cases, a functional goal can be achieved by using more than one problem types. The choice of the problem type to be used depends on many factors that include runtime constraints, underlying software and hardware infrastructure, etc. For example, customer churn analysis is a technique for predicting the customers that are most likely to leave the current service provider (retail, telecom or financial) for a competitor. This analysis can use one of the three problem types: inferential statistics, supervised learning or unstructured data analysis. One approach models individual customer's behavior using various parameters such as duration of service, user transaction history, etc. These parameters are then fed either to a statistical model such as regression or to a supervised learning model such as a decision tree, to predict if a customer is likely to defect [21]. The second approach, models behavior of a customer based on her interactions with other customers. This strategy is commonly used in the telecom sector, where customer calling patterns are used to model subscriber relationships as a graph. This unstructured graph can then be analyzed to identify subscriber groups and their influential leaders- usually the well-connected and most active subscribers. These leaders can then be targeted for marketing campaigns to reduce defection in the members of her group [22].

The library component is usually designed to be portable and broadly applicable across multiple analytic solutions (e.g., the DeepQA runtime that powers the Watson system). A library usually provides implementations of specific models of the common problem types. For example, an unsupervised learning problem can be solved using one of many models including associative mining, classification, or clustering [14]. Each model can, in turn, use one or more algorithms for its implementation. For instance, the associative mining model can be implemented using the different associative rule mining or decision tree algorithms. Similarly, classification can be implemented using nearest-neighbor, neural network, or naive Bayes algorithms. It should be noted that in practice, the separation between models and algorithms is not strict and many times, an algorithm can be used for supporting more than one models. For instance, neural networks can be used for clustering or classification. Finally, depending on how the problem is formulated, each algorithm uses specific data structures and kernels. For example, many algorithms formulate the problem using dense or sparse matrices and invoke kernels like matrix-matrix and matrix-vector multiplication, matrix factorization, and linear system solvers. These kernels are sometimes optimized for the underlying system architecture, in form of libraries such as IBM ESSL [15] or Intel MKL [17]. Any kernel implementation can be characterized according to how it manages parallel execution, if at all, and how it manages data and maps it to the system mem-

[1] We have expanded the classification proposed by Davenport et al. [8, 9].

Analytical applications	Functional goals	Problem types
Supply chain management, Product scheduling, Logistics, Routing, Workforce management	Prescription	Optimization
Revenue prediction, Disease spread prediction, Semiconductor yield analysis, Predictive policing	Prediction	Unsupervised/Supervised learning Descriptive/Inferential statistics
Retail sales analysis, Financial reporting, Budgeting, System management analysis, Social network analysis	Reporting	Structured/Unstructured data analysis
VLSI sensitivity analysis, Insurance risk modeling, Credit risk analysis, Physics/Biology simulations, Games	Simulation	Modeling and simulation Descriptive/Inferential statistics
Topic/Sentiment analysis, Computational chemistry, Document management, Searching, Bio- and Chemo-informatics	Pattern matching	Structured/Unstructured data analysis Unsupervised/Supervised learning
Cross-sale analysis, Customer retention, Music/Video and restaurant recommendation, Intrusion detection	Recommendation	Unsupervised/Supervised Learning Structured/Unstructured data analysis
Web-traffic analysis, Fraud detection, Geological Sensor networks, Geographical analytics (Maps)	Alerting	Descriptive/Inferential statistics Unsupervised/Supervised learning
Customer relationship analysis, Weather forecasting, Medical Informatics, Econometrics, Computational finance	Quantitative analysis	Descriptive/Inferential statistics Unsupervised/Supervised learning

Table 2. Examples of analytics applications, associated functional goals, and analytical problem types

ory and I/O architecture. Many parallel kernels can use shared or distributed memory parallelism. In particular, if the algorithm is embarrassingly parallel, requires large data, and the kernel is executing on a distributed system, it can often use the MapReduce approach [10]. At the lowest level, the kernel implementation can often exploit hardware-specific features such as short-vector data parallelism (SIMD) or task parallelism on multi-core CPUs, massive data parallelism on GPUs, and application-specific parallelism using Field-programmable gate arrays (FPGAs).

3 Analytics Exemplars

Given the wide variety of algorithmic and system alternatives for executing analytics applications, it is difficult for solution developers to make the right choices to address specific performance issues. To alleviate this problem, we have analyzed the functional flow (Figure 2) of a wide set of key applications across multiple analytics domains and have isolated repeated patterns in analytical applications, algorithms, data structures, and data types. At IBM, we use this information to optimize analytic applications and libraries for modern system structures and to evolve and in some cases specialize our processor and system designs to better suit analytic applications.

Towards this goal, we have identified a set of widely-used analytical models that capture the most important computation and data access patterns of the analytics applications that we have studied [5, 25]. These models, referred to as *Analytics Exemplars*, cover the prevalent analytical problem types and each exemplar can be used to address one or more functional goals. Table 3 presents the list of thirteen exemplars, along with target functional goals and key algorithms used for implementing these exemplars. As Table 3 illustrates, each exemplar can be implemented by one or more distinct algorithms. Some of the algorithms can be used for implementing more than one exemplars, e.g., the Naive Bayes algorithm can be used in text analytics and for general clustering purposes. Each algorithm, depending on the runtime constraints, i.e., whether the application data can fit into main memory or not, can use a variety of algorithmic kernels (Figure 2). For more details on the algorithms and their implementations, the reader is referred to [5, 34].

Table 4 presents a summary of computational patterns, key data types, data structures and functions used by algorithms for each exemplar. As Table 4 illustrates, while different exemplars demonstrate distinct computational and runtime characteristics, they also exhibit key similirities. Broadly, the analytic exemplars can be classified into two classes: the first class exploits linear-algebraic formulations and the second uses non-numeric data structures (e.g., hash tables, trees, bit-vectors, etc.). Exemplars belonging to the first class, e.g., Mathematical Programming, Regression Analysis, and Neural Networks, operate primarily on matrices and vectors. Matrices are either sparse or dense, and are used in various linear algebraic kernels like the matrix multiplication, inversion, transpose, and factorization. The second class, which includes clustering, nearest-neighbor search, associative rule mining, decision tree learning, use data structures like hash-tables, queues, graphs, and trees, and operate on them using set-oriented, probabilistic, graph-traversal, or dynamic programming algorithms. Exemplars like mathematical programming, text analytics, and graph analytics can use either of these approaches. The analytic exemplars use a variety of types, such as Integers, Strings, Bit-vector, and single and double precision floats, to represent the application data. This information is then processed using different functions that compare, transform, and modify input data. Examples of common analytic functions include various distance functions (e.g., Euclidian), kernel functions (e.g., Linear, Sigmoid), aggregation functions (e.g., Sum), and Smoothing functions (e.g., correlation). These functions, in turn, make use of intrinsic library functions such as `log`, `sine` or `sqrt`.

Table 5 summarizes the runtime characteristics of the analytics exemplars. The key distinguishing feature of analytics applications is that they usually process input data in read-only mode. The input data can be scalar, structured with one or more dimensions, or unstructured, and is usually read from files, streams or relational tables in the binary or text format. In most cases, the input data is large, which requires analytics applications to store and process data from disk. Notable exceptions to this pattern are Monte Carlo Methods and Mathematical Programming, which are inherently in-memory as they operate on small input data. The results of analysis are usually smaller than the input data. Only two exemplars, association rule mining and on-line analytical processing (OLAP) generate larger output. Finally, analytics applications can involve one or more stages (real-time execution can be considered to have only one stage), where each stage invokes the corresponding algorithm in an iterative or non-iterative manner. For the iterative workloads, for the same input data size, the running time can vary depending on the precision required in the results.

Model Exemplar (Problem type)	Functional goals	Key algorithms
Regression analysis (Inferential statistics)	Prediction, Quantitative analysis	Linear, Non-linear, Logistic, and Probit regression
Clustering (Supervised learning)	Pattern matching, Recommendation, Prediction, Reporting	K-Means and Hierarchical clustering Expectation-Maximization Clustering, Naive Bayes
Nearest-neighbor search (Unsupervised learning)	Pattern recognition, Prediction, Recommendation	K-d, Ball, and Metric trees, Approximate Nearest-neighbor Locality-sensitive Hashing, Kohonen networks
Association rule mining (Unsupervised learning)	Recommendation	Apriori, Partition, FP-Growth, Eclat and MaxClique, Decision trees
Neural networks (Supervised learning)	Prediction, Pattern matching	Single- and Multi-level perceptrons, Radial-Basis Function (RBF), Recurrent, and Kohonen networks
Support Vector Machines (SVMs) (Supervised learning)	Prediction, Pattern matching	SVMs with Linear, Polynomial, RBF, Sigmoid, and String kernels
Decision tree learning (Supervised learning)	Prediction, Recommendation	ID3/C4.5, CART, CHAID, QUEST
Time series processing (Structured/Unstructured data analysis)	Prediction, Pattern matching, Reporting, Alerting	Trend, Seasonality, Spectral analysis, ARIMA, Exponential smoothing
Text analytics (Structured/Unstructured data analysis)	Pattern matching, Reporting	Naive Bayes classifier, Latent semantic analysis, String-kernel SVMs, Non-negative matrix factorization
Monte Carlo methods (Modeling and simulation)	Simulation, Quantitative analysis	Markov-chain, Quasi-Monte Carlo methods
Mathematical programming (Optimization)	Prescription, Quantitative analysis	Primal-dual interior point, Branch & Bound methods, Traveling salesman, A* algorithm, Quadratic programming
On-line analytical processing (OLAP) (Structured data analysis)	Reporting, Prediction	Group-By, Slice_and_Dice, Pivoting, Rollup and Drill-down, Cube
Graph analytics (Unstructured data analysis)	Pattern matching, Reporting, Recommendation	Eigenvector Centrality (e.g., PageRank), Routing, Coloring, Searching and flow algorithms, Clique and motif finding

Table 3. Analytics exemplar models, along problem types and key application domains

Model Exemplar	Computational pattern	Key data types, Data structures, and Functions
Regression Analysis	Matrix inversion, LU decomposition, Transpose Cholesky factorization	Double-precision and Complex data Sparse/dense matrices, Vectors
Clustering	Metric-based iterative convergence	Height-balanced tree, Graph, Distance functions (Euclidean, Manhattan, Minkowski, and Log-Likelihood), `log` functions
Nearest-Neighbor Search	Non-iterative distance calculations via metric functions Singular value decomposition, Hashing	Higher-dimensional data structures (k-d, and Metric trees), Hash tables, Euclidean and Hamming distance functions
Association Rule Mining	Set intersections, Unions, and Counting	Hash-tree, Relational tables, Prefix trees, Bit vectors
Neural Networks	Iterative Weighted Feedback networks Matrix multiplication, Inversion, Cholesky factorization	Sparse/dense matrices, Vectors, Double-precision/Complex data Gaussian, Multiquadric, Spline, Logistic, Smoothing functions
Support Vector Machines	Cholesky factorization, Matrix multiplication	Double-precision floats, Sparse matrices, Vectors Kernel functions (e.g., Linear, Sigmoid, Polynomial, String)
Decision Tree Learning	Dynamic programming, Recursive tree operations	Integers, Double-precision floats, Trees, Vectors, `log` function
Time Series Processing	Smoothing via averaging, Correlation Fourier and Wavelet transforms	Integers, Single-/Double-precision floats, Dense matrices, Vectors `cosine`, `sine`, `log` functions, Distance and Smoothing functions
Text Analytics	Parsing, Bayesian modeling, String matching Hashing, Singular value decomposition Matrix multiplication, Transpose, Factorization	Integers, Single/Double precision, Characters, Strings Sparse matrices, Vectors, Inverse indexes, String functions, Distance functions
Monte Carlo Methods	Random number generators (e.g., Mersenne, Gaussian) Polynomial evaluation, Interpolation	Double-precision floats, Bit vectors Bit-level operations (shift, mask), `log`, `sqrt` functions
Mathematical Programming	Matrix multiplication, Inversion, Cholesky factorization Dynamic programming, Greedy algorithms, Backtracking-based search	Integers, Double-precision floats, Sparse Matrices, Vectors, Trees, Graphs
On-line Analytical Processing	Grouping and ordering multi-dimensionsal elements Aggregation over hierarchies	Prefix trees, Relational tables, OLAP Operators (e.g., `CUBE`), Strings Sorting, Ordering, Aggregation operators, e.g., Sum or Average
Graph Analytics	Graph traversal, Eigensolvers, Matrix-vector and Matrix-matrix multiplication, Factorization	Integer, Single-/Double-precision floats, Adjacency/incident lists Trees, Queues, Dense/Sparse matrices

Table 4. Computational characteristics of the analytics exemplars

Model Exemplar	Execution characteristics		Input-Output characteristics	
	Methodology	Memory Issues	**(Read-only)** Input Data	Output Data
Regression Analysis	Iterative	In-memory Disk-based	Large historical Structured	Small Scalar
Clustering	Iterative	In-memory Disk-based	Large historical Unstructured or structured	Small scalar Unstructured or structured
Nearest-Neighbor Search	Non-iterative	In-memory	Large historical Structured	Small Scalar or structured
Association Rule Mining	Iterative Non-iterative	In-memory Disk-based	Large historical Structured	**Larger** Structured
Neural Networks	Iterative Two Stages	In-memory Disk-based	Large Structured	Small Scalar
Support Vector Machines	Iterative Two Stages	In-memory Disk-based	Large Structured	Small Scalar
Decision Tree Learning	Iterative Two Stages	In-memory Disk-based	Large Structured & Unstructured	Small Scalar
Time Series Processing	Non-iterative Real-time	In-memory	High volume streaming Structured or unstructured	Small scalar or streaming Structured or unstructured
Text Analytics	Iterative Non-iterative	In-memory Disk-based	Large historical or streaming Structured or unstructured	Large or small Structured or unstructured
Monte Carlo Methods	Iterative	In-memory	Small Scalar	Large Scalar
Mathematical Programming	Iterative	In-memory	Small Scalar	Small Scalar
On-line Analytical Processing	Non-iterative	In-memory Disk-based	Large historical Structured	**Larger** Structured
Graph Analytics	Iterative	In-memory Disk-based	Large historical Unstructured	Small Scalar or unstructured

Table 5. Runtime characteristics of the analytics exemplars

4 System Implications

Given the varied computational and runtime characteristics of the analytics exemplars, it is clear that a single systems solution for different analytics applications would be sub-optimal. As Tables 4 and 5 demonstrate, each exemplar has a unique set of computational and runtime features, and ideally, every exemplar would get a system tailor-made to match its requirements. However, we have also observed that different analytic exemplars share many computational and runtime features. Therefore, for a systems designer, the challenge is to customize analytics systems using as many re-usable software and hardware components as possible.

Table 6 describes system opportunities for accelerating analytics exemplars. Based on the computational and runtime characteristics described in Tables 4 and 5, we first identify key bottlenecks in the execution of analytic exemplars, namely compute-bound, memory-bound, and I/O bound (which covers both disk and network data traffic). As Table 6 illustrates, a majority of the analytics exemplars are compute bound in the in-memory mode and I/O-bound when in the disk-based mode. The compute-bound exemplars can benefit from traditional task-based parallelization approaches on multi-core processors, as well as by hardware-based acceleration via SIMD instructions or using GPUs. When used in the disk-based scenarios, these exemplars can improve their I/O performance by using solid state drives or data compression. Some of the analytics exemplars are memory-bound due to their reliance on algorithms that traverse large in-memory data structures such as trees or sparse matrices. For these exemplars, a better memory sub-system, with faster, larger, and deeper memory hierarchies, would be most beneficial. Once the memory accesses are optimized, these exemplars can also benefit from traditional computational acceleration techniques. Finally, some of the exemplars exhibit unique computational patterns (e.g., bit-level manipulations, pattern matching, or string processing) which could be accelerated using special-purpose processors such as FPGAs or by introducing new instructions in general-purpose processors. In most cases, the exemplars can be accelerated using commodity hardware components (e.g., multi-core processors, GPUs or SSDs). These hardware components can be then used to optimize re-usable software kernel functions (e.g., numerical linear algebra, distance functions, etc.), which themselves can be parallelized by a variety of parallelization techniques such as task parallelism, distributed-memory message-passing parellelism or MapReduce [24, 2]. These functions be used as a basis of specialized implementations of the exemplars. Such hardware-software co-design enables optimized analytics solutions that can balance customization and commoditization.

An example of hardware-software co-design is the Netezza data warehouse and analytics appliance. The Netezza appliance supports both SQL-based OLAP and analytics queries. Netezza uses a combination of FPGA-based acceleration and customized software to optimize data-intensive mixed database and analytics workloads with concurrent queries from thousands of users. The Netezza system uses two key principles to achieve scalable performance: (1) Reduce unnecessary data traffic by moving processing closer to the data, and (2) Use parallelization techniques to improve the processing costs. A Netezza appliance is a distributed-memory system with a host server connected to a cluster of independent servers called the snippet blades (S-Blades). A Netezza host first compiles a query using a cost-based query optimizer that uses the data and query statistics, along with disk, processing, and networking costs to generate plans that minimize disk I/O and data movement. The query compiler generates executable code segments, called snippets which are executed in parallel by S-blades. Each S-blade is a self-contained system with multiple multi-core CPUs, FPGAs, gigabytes of memory, and a local disk subsystem. For a snippet, the S-Blade first reads the data from disks into memory using a technique to reduce disk scans. The data streams are then processed by FPGAs at wire speed. In a majority of cases, the FPGAs filter data from the origi-

Model Exemplar	Bottleneck	Acceleration requirements and opportunities
Regression Analysis	Compute-bound I/O-bound	Shared- and Distributed-memory task parallelism, Data parallelism via SIMD or GPUs Faster I/O using solid state drives
Clustering		
Nearest-Neighbor Search		
Neural Networks		
Support Vector Machines		
Association Rule Mining	I/O-bound	Shared-memory task parallelism, Faster I/O using solid state drives Faster bit operations or tree traversals via FPGAs
Decision Tree Learning	Memory-bound	Larger and deeper memory hierarchies, Data parallelism via SIMD
Time Series Processing	Compute-bound Memory-bound	Shared- and Distributed-memory task parallelism, Data parallelism via SIMD or GPUs High-bandwidth, low-latency memory subsystem, Pattern matching via FPGA
Text Analytics	Memory-bound I/O-bound	Shared- and Distributed-memory task parallelism, Data parallelism via SIMD or GPUs Larger and deeper memory hierarchies, Faster I/O via solid state drives Pattern matching and string processing via FPGA
Monte Carlo Methods	Compute-bound	Shared- and Distributed-memory task parallelism, Data parallelism via SIMD or GPUs Faster bit manipulations using FPGAs or ASICs
Mathematical Programming	Compute-bound	Shared-memory task parallelism, Massive data-parallelism via GPUs Larger and deeper memory hierarchies, Search-tree traversals via FPGAs
On-line Analytical Processing	Memory-bound I/O-bound	Shared- and Distributed-memory task parallelism, Data parallelism via SIMD or GPUs Larger and deeper memory hierarchies, Pattern Matching via FPGAs, Faster I/O using solid state drives
Graph Analytics	Memory-bound	Shared-memory task parallelism, Larger and deeper memory hierarchies

Table 6. Opportunities for parallelizing and accelerating analytics exemplars

nal stream, and only a tiny fraction is sent to the S-Blade CPUs for further processing. The FPGAs can also execute some additional functions which include decompression, concurrency control, projections, and restrictions. The CPUs then execute either database operations like sort, join, or aggregation or core mathematical kernels of analytics applications on the filtered data streams. Results from the snippet executions are then combined to compute the final result. The Netezza architecture also supports key data mining and machine learning algorithms on numerical data (e.g., matrices) stored in relational tables.

A key lesson learned from the design of Netezza has been the huge value of specializing system design for analytics. Orders of magnitude improvements in efficiency can be achieved by carefully analyzing the system requirements and innovating using a collaborative software-hardware design methodology.

5 Summary

In this survey paper and the accompanying research report [5], we have reviewed the growing field of analytics that uses mathematical formulations to solve business and consumer problems. We have identified some of the key techniques employed in analytics, called *analytics exemplars*, both to serve as an introduction for the non-specialist and to explore the opportunity for greater optimization for parallel computer architectures and systems software. We hope this work spurs follow-on work on analyzing and optimizing analytics workloads.

6 References

[1] Armacost, A. P., Barnhart, C., Ware, K. A., and Wilson, A. M. UPS Optimizes Its Air Network. *Interfaces 34*, 1 (January-February 2004).

[2] Bekkerman, R., Bilenko, M., and Langford, J., Eds. *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011. To appear.

[3] Bell, R. M., Koren, Y., and Volinsky, C. All Together Now: A Perspective on the Netflix Prize. *Chance 23*, 1 (2010), 24–29.

[4] Bhattacharya, I., Godbole, S., Gupta, A., Verma, A., Achtermann, J., and English, K. Enabling analysts in managed services for CRM analytics. In *In Procs. of the 2009 ACM KDD Intl. Conf. on Knowledge and Data Discovery* (2009).

[5] Bordawekar, R., Blainey, B., Apte, C., and McRoberts, M. Analyzing analytics, part 1: A survey of business analytics models and algorithms. Tech. Rep. RC25186, IBM T. J. Watson Research Center, July 2011.

[6] Cantor, R., Packer, F., and Cole, K. Split ratings and the pricing of credit risk. Tech. Rep. Research Paper No. 9711, Federal Reserve Bank of New York, March 1997.

[7] Crosbie, P., and Bohn, J. Modeling default risk, December 2003. Moody's KMV.

[8] Davenport, T., and Harris, J. *Competing on Analytics, The New Science of Winning*. Harvard Business School Press, 2007.

[9] Davenport, T., Harris, J., and Morison, R. *Analytics at Work, Smarter Decisions, Better Results*. Harvard Business School Press, 2010.

[10] Dean, J., and Ghemawat, S. Mapreduce: a flexible data processing tool. *Communications of the ACM 53*, 1 (2010).

[11] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. Building Watson: An Overview of the DeepQA Project. *AI Magazine 59*, Fall (2010).

[12] Go, A., Bhayani, R., and Huang, L. Twitter sentiment classification using distant supervision. Tech. rep., Computer Science Department, Stanford University, December 2009.

[13] Goode, E. Sending the police before there's a crime. The New York Times, August 16, 2011.

[14] Han, J., and Kamber, M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2006.

[15] IBM CORP. Engineering and Scientific Subroutine Library (ESSL) and Parallel ESSL. `www.ibm.com/systems/software/essl`.

[16] IBM NETEZZA. Netezza Data Warehouse Appliances. `www.netezza.com`.

[17] INTEL INC. Intel Math Kernel Library. `software.intel.com`.

[18] JOYCE, J. Pandora and the Music Genome Project. *Scientific Computing 23*, 10 (September 2006), 40–41.

[19] LOHATEPANONT, M., AND BARNHART, C. Airline Schedule Planning: Integrated Models and Algorithms for Schedule Design and Fleet Assignment. *Transportation Science 38*, 1 (February 2004).

[20] MOHLER, G., SHORT, M., BRANTINGHAM, P., SCHOENBERG, F., AND TITA, G. Self-exciting point process modeling of crime. *Journal of American Statistical Association 106*, 493 (March 2011).

[21] MUTANEN, T. Customer churn analysis- a case study. Tech. Rep. VTT-R-01184-06, Technical Research Centre of Finland, 2006.

[22] NANAVATI, A. A., GURUMURTHY, S., DAS, G., CHAKRABORTY, D., DASGUPTA, K., MUKHERJEA, S., AND JOSHI, A. On the structural properties of massive telecom call graphs: Findings and implications. In *Proc. of the Conference on Information and Knowledge Management (CIKM'06)* (November 2006), pp. 435–444.

[23] NGAI, E. W. T., XIU, L., AND CHAU, D. C. K. Application of data mining techniques in customer relationship management: A literature review and classification. *Expert Systems with Applications 36* (2009), 2592–2602.

[24] RAJARAMAN, A., AND ULLMAN, J. *Mining Massive Datasets*. Cambridge University Press, 2010. Free version available at `infolab.stanford.edu/~ullman/mmds.html`.

[25] REXER, K., ALLEN, H., AND GEARAN, P. 2010 data miner survey summary. In *Procs. of the Predictive Analytics World* (October 2010).

[26] RICHTER, Y., YOM-TOV, E., AND SLONIM, N. Predicting customer churn in mobile networks through analysis of social groups. In *Procs. of the SIAM International Conference on Data Mining, SDM 2010* (2010), pp. 732–741.

[27] SCIENCE SPECIAL ISSUE. Dealing with data. *Science 331*, 6018 (February 2011).

[28] SINDHWANI, V., GHOTING, A., TING, E., AND LAWRENCE, R. Extracting insights from social media with large-scale matrix approximations. *IBM Journal of Research and Development 55*, 5 (Sept-Oct 2011), 9:1–9:13.

[29] SONY PICTURES INC. Jeopardy! The IBM Challenge. `www.jeopardy.com/minisites/watson`.

[30] SPLUNK INC. Splunk Tutorial, 2011. `www.splunk.com`.

[31] SUMAN, A. Automated face recognition, applications within law enforcement: Market and technology review, October 2006.

[32] THE ECONOMIST. Algorithms: Business by numbers. Print Edition, 13th September, 2007.

[33] THE ECONOMIST. Data, data everywhere. Print Edition, 25th February, 2010.

[34] WU, X., KUMAR, V., QUINLAN, J. R., GHOSH, J., YANG, Q., MOTODA, H., MCLACHLAN, G. J., NG, A., LIU, B., YU, P. S., ZHOU, Z.-H., STEINBACH, M., DAVID J, H., AND STEINBERG, D. Top 10 algorithms in data mining. *Knowledge Information Systems 14* (2008), 1–37.