Big Data Optimization Within Real World Monitoring Constraints

Kristian Helmholt and Bram van der Waaij

Abstract Large scale monitoring systems can provide information to decision makers. As the available measurement data grows, the need for available and reliable interpretation also grows. To this, as decision makers require the timely arrival of information, the need for high performance interpretation of measurement data also grows. Big Data optimization techniques can enable designers and engineers to realize large scale monitoring systems in real life, by allowing these systems to comply to real world constrains in the area of performance, reliability and reliability. Using several examples of real world monitoring systems this chapter discusses different approaches in optimization: data, analysis, system architecture and goal oriented optimization.

Keywords Measurement · Decision · Monitoring · Constraint-based optimization

1 Introduction

Monitoring systems enable people to respond adequately to changes in their environment. Big Data optimization techniques can play an important role in realization of large scale monitoring systems. This chapter describes the relationship between Big Data optimization techniques and real world monitoring in terms of optimization approaches that enable monitoring systems to satisfy real world deployment constraints.

K. Helmholt (✉) · B. van der Waaij
Netherlands Organisation for Applied Scientific Research (TNO),
Groningen, The Netherlands
e-mail: kristian.helmholt@tno.nl

B. van der Waaij
e-mail: bram.vanderwaaij@tno.nl

A. Emrouznejad (ed.), *Big Data Optimization: Recent Developments and Challenges*, Studies in Big Data 18, DOI 10.1007/978-3-319-30265-2_11

This chapter starts with a high level overview of the concept of monitoring in Sect. 2. After that the relationship between large scale monitoring systems and Big Data is described in Sect. 3. This is done using constraints on monitoring systems with respect to performance, availability and reliability, which are critical success factors for monitoring systems in general and also have a strong relationship with Big Data. Then, in Sect. 4, several solution approaches from the field of Big Data optimization are presented for designers and engineers that need to stay within constraints as set forward by the context of the deployment of a monitoring system. Also, the impact on several other constraints is taken into account. This chapter ends with Sect. 5 presenting conclusions.

2 Monitoring

A thorough understanding of the relationship between Big Data optimization and monitoring, starts with a global understanding of the concept of monitoring. This understanding provides a means to comprehend the Big Data related constraints put upon monitoring by the real world, which will be described in the next section. In this section an understanding of monitoring systems will be provided by describing a real world example.

2.1 General Definition

The word 'monitor' is supposed to be derived from the Latin monitor ("warner"), related to the Latin verb monere ("to warn, admonish, remind"). A monitoring system could therefore be defined as a collection of components that interact in order to provide people and/or other systems with a warning with respect to the state of another object or system. A (very) small scale example of a monitoring system is a smoke detector: it continuously measures the visibility of the surrounding air. Once a threshold level with respect to that visibility has been crossed, an alarm is sounded. In this chapter, the definition of a monitoring system is extended to systems that are not primarily targeted at warning, but possibly also at learning. This is because modern day monitoring systems can and need to adapt to changes in the environment they monitor. This means that modern monitoring systems can—for example—also be used to find the relationship between behavior of different objects and/or parameters in the system they observe. So, in turn the definition of a monitoring system in this chapter becomes:

> *"a collection of components that interact in order to provide people and/or other systems with information with respect to the state of a system of other objects, based on (real-time) measurement data on that system (of objects)."*

In this chapter the scope is limited to large scale monitoring systems. In the remainder of this chapter the abbreviation LSMS is used. These systems collect vast amounts of (measurement) data—using sensors—on the real world and process it to information, which could then be used to decide upon or to learn from. As will become clear in the next paragraphs, LSMSs can be used for many different systems. For example, a dike LSMS can be used to monitor dikes for failure (which would result in flooding). Another example is an LSMS that monitors the health and wellbeing of hundreds of thousands of cows throughout their lifetime.

In order to structure the material in this chapter, a simple but effective monitoring framework is used to position functions of a monitoring system. This framework—depicted in Fig. 1—enables describing the relationship with Big Data optimization later on in this chapter. The basic idea of this framework is that in all monitoring systems three generic steps can be distinguished. Using the relatively simple example of a radar the following three steps can be described:

1. **Measurement**: *mapping a certain aspect of reality to a unit of measurement of a physical magnitude (or quantity) using a sensor*. For example a radar that sends out radio waves and receives reflections which tells something about distances from the object to the radar antenna.
2. **Interpretation**: *interpretation of sensor data into information—using expert models—about an* ***object*** *or system to be monitored*. For example interpretation of radar data into information about an object that is flying towards a radar.
3. **Decision to inform**: *applying some kind of rule set to determine if a human or other system should be informed (e.g. warned)*. For example, based on the speed of an object flying towards the radar, a warning could be produced if an object is coming into fast or too close.

Note that there are other generic and more elaborate descriptions of monitoring (& control) systems, like the famous Observe, Orient, Decide, and Act (often abbreviated as OODA) loop [1] or the 'Knowledge Discovery' as described in [2]. These shall not be discussed in this chapter, since this simple decomposition in Measurement, Interpretation and Decision (MID) is only needed as a means to position monitoring constraints from a Big Data optimization perspective later on. Also, control systems are beyond the scope of this chapter.

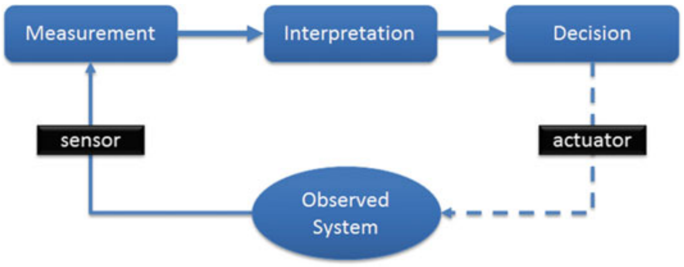

Fig. 1 Measurement, interpretation, decision framework

This concludes the very high level description of monitoring in general. With this in mind, an example of an LSMS is presented in the next section. This example will be used to describe specific Big Data related aspects in the section on Big Data related constraints.

2.2 Dike Monitoring Example

In the Netherlands dikes prevent more than half of the population of 17 million people from losing their houses, losing their livelihood and ultimately drowning. Construction and maintenance of dikes is relatively expensive, because other than safety, dikes provide no direct financial output, like a factory that produces products that can be sold. Note that the Netherlands there is more than 3500 km of primary dikes and more than 10,000 km of secondary dikes. From both a safety as well as economic point of view, it is valuable to both know how and when a dike could fail. From a socio-economic point of view dikes should be safe enough, but too much over dimensioning is a waste of money. Since this could be spent elsewhere, for example in healthcare where lives are also at stake.

Several years ago, it seemed appropriate to a consortium of water boards (i.e. local/regional governments targeted at water management), research organizations and industrial partners to start working on an LSMS for dikes [3, 4]. Advances in computer science, geotechnical sciences and the decline of costs of computation, communication and storage hardware, made it seem as if there could be a business case for dike monitoring. In other words the expected costs of the LSMS seemed to be less than the avoided costs in case of inefficient dike maintenance (too little or

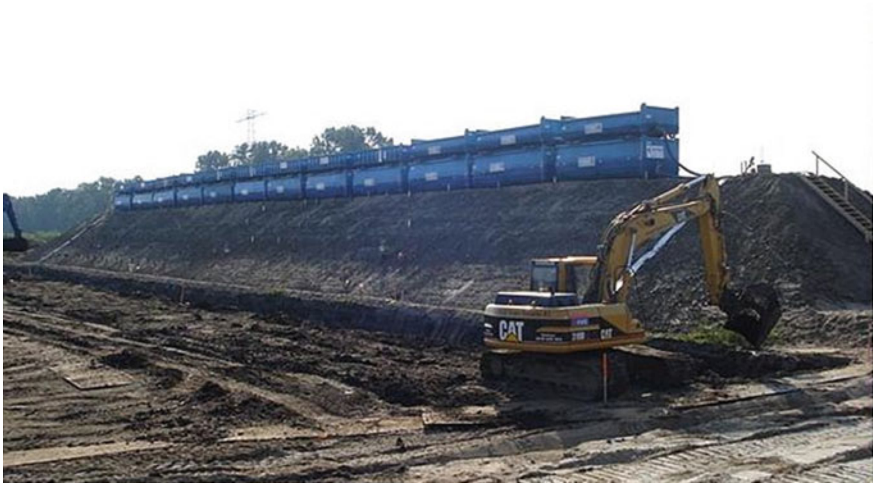

Fig. 2 Dike being monitored with an large scale monitoring system

too much). The result of the collaboration was the creation of a test facility for the monitoring of dikes (Fig. 2). Several experimental dikes have been constructed there, which contained many different types of sensors [3].

Measurement. A type of sensor used that produced large amounts of data is fiberglass cloth, which can be used to measure the shape of a dike, by wrapping it onto the dike. When the dike changes shape, the fiberglass cloth bends resulting in different measurement data. Another type of sensor is the ShapeAccelArray/Field (SAAF), a string of sensors that provides (relative) position, acceleration and orientation information. Like fiberglass sensors, a SAAF can provide a vast amount of data in a short amount of time. For example a 3 m SAAF can contain 10 measurement points, that each can provide 9 measurement values. Providing 10 * 9 values in total every 5 s. A final example for determining the shape of a dike is to use remote sensing such as satellite data [5].

Interpretation. The end-users of the dike LSMS are interested in the stability of the dike: what is the chance that it will fail. There is no physical instrument that measures this abstract ‘unit of measurement’, it has to be calculated according to a computational model that all LSMS involved parties agree upon. This model must take into account significantly contributing to the dike stability. For example the geotechnical make-up of the dike and the (expected) forces acting upon the dike, like water levels on both sides. This can be done without using sensor data and by assuming possible (extreme) parameter values for the models involved and calculate the likelihood of failure. However, while progress has been made in the last decennia with theoretical models for dike behavior, uncertainty remained. For this test facility dike stability models have therefor been adapted to use measurements from sensors inside or targeted at the dike. During the experiments at the facility, sensor developers could find out if the data from their (new) sensors contributed to reducing uncertainty about dike failure. Geotechnical model builders could find out if their models were using data from sensor efficiently. Hence the name of the test facility, which was ‘IJkdijk’, as ‘ijk’ means ‘to calibrate’ in Dutch.

Decision. A dike LSMS is an excellent example of the need for a monitoring system that produces information on which people can rely. When an LSMS warns that a dike will fail within the next two days, people are almost forced to start evacuating if there is no possibility to strength the dike anymore. If it turns out the LSMS was wrong (i.e. the dike would not have failed), the impact of this false alarm is huge from a socio-economic point of view. When an LSMS does not warn, while the dike is about to fail, resulting in a flood nobody was warned about, the socio-economic impact is also very huge. So in short: if society wants to trust LSMSs for dikes there must be no error or failure in the chain of components that transform sensor data into decision information.

As shown by the dike LSMS, there can be strong requirements to an LSMS system that act as heavy constraints, which are also related to Big Data optimization, as this can influence the extent to which an LSMS can meet the requirements. These will be covered in the next section.

3 Big Data Related Constraints to Monitoring

In order to be economically sustainably deployed in the real world there are certain requirements a LSMS must meet. Designers and engineers of an LSMS that have to come up with a technical solution, consider these requirements as constraints to their solution. An obvious constraint is that the financial value of the information produced by the LSMS should not exceed the costs of the LSMS itself. This includes costs of hardware, energy, labor, etc. In this chapter the focus is on three types of constraints (performance, availability and reliability) that are both strongly related to the added value of a monitoring system to society, as well as Big Data optimization. It could be argued that there are other such constraints, but for reasons of scope and size of this chapter it has been decided not to include those.

In order to describe the relationship between Big Data and LSMSs from the viewpoint of constraints in an efficient way, this relationship will first be described from the viewpoint of data collection and interpretation. After describing the relationship from constraints, solution approaches for keeping within the constraints will be presented in the next section.

3.1 The Big Data in Monitoring

The relevance of Big Data optimization to LSMS depends—to a large extent—on the way data is collected and interpreted. In this subsection this relationship will be explored by looking at different aspects of data collection and interpretation in LSMSs.

3.1.1 Abstraction Level of Interpreted Information

In the dike LSMS example the LSMS has to provide information on ‘dike stability’. This is an abstract concept and cannot directly be measured and has to be interpreted. The value of this interpreted ‘unit of measurement’ can only be produced by using a computational model that computes it, based on sensor measurements and a model of the dike. This can result in much computational effort, even if there are relatively few sensors installed.

The higher the abstraction level of a interpreted ‘unit of measurement’ is the more (sub)models tend to be required. This is illustrated by an example of an LSMSs for underground pipelines illustrates this [6]. This LSMS determines the chance of failure of a segment of underground pipeline and is depicted in Fig. 3. The interpretation uses based on:

1. Actual soil movements, measured by underground position sensors.
2. Expected soil movements, based on soil behavior models, where (expected) external forces onto the soil are also taken into account.

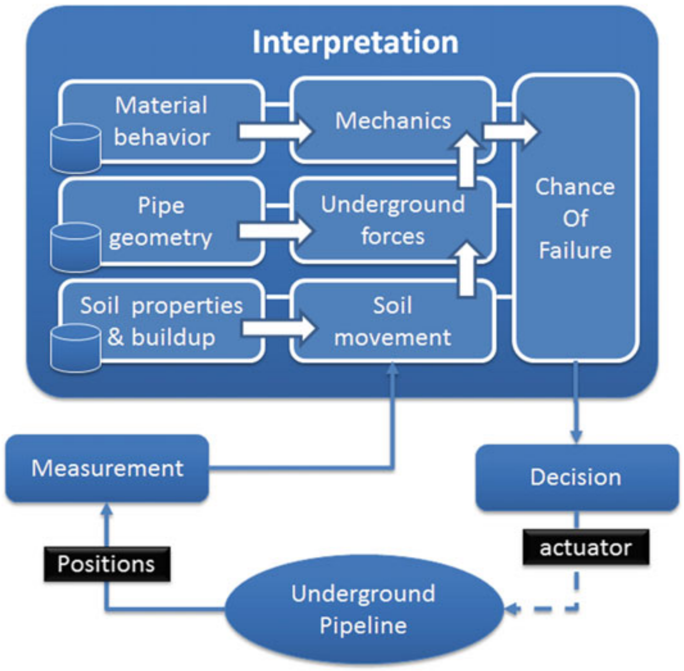

Fig. 3 Example of interpretation using model with a high level of abstraction

3. Estimated underground geometry of pipeline structures, based on construction and maintenance plans.
4. Estimated mechanical properties of pipeline segments and structures, based on physics and analysis of similar pipeline segments elsewhere.

The LSMS uses to the actual ground movements and the assumed model of the underground to create a better estimate of the actual underground. It then computes forces on the pipe structure and uses those in combination with estimated mechanical properties to determine an estimate of the structural reliability. Note that it does also take into account possible variations of parameters involved, due to the uncertainty.

Each of these different aspects (e.g. soil movement, structural behavior of pipe geometry based on ground forces) requires a model of its own. Next to sensor data, it also requires data on the ground buildup and geometries of pipelines. Because the ground buildup tends to vary, just as the pipeline geometries, estimating the chance of failure for all segments of—for example a country like—the Netherlands results in an explosion of computations to be carried out on a large dataset.

3.1.2 Temporal Issues

The way data has to be collected and interpreted (also) depends on what LSMS (end-)users expect the LSMS to do, from a temporal point of view:

1. assessment of the **current state** of the system under observation
2. an estimation of **possible future states** of that system

As was shown above in the case of the LSMS for underground pipelines, assessing the current state of a system under observation can already require many computations because of the need to evaluate different models involved. When information on possible future states is also required, there is need for even more computation and possibly extra storage. This is due to the fact that the interpretation part of the monitoring system needs a model of the system under observation, that allows to predict or estimate future states of the system. Roughly said, two types of models can be distinguished.

Descriptive model: the inner workings of a system are understood to a certain extent, and based on the application of the model to the measurement data it can be predicted how the system will behave next (within a certain band of uncertainty). For example: if you measure the time, you can use models of the earth's rotation around the sun to determine your position in space, relative to the sun.

Phenomenological model: the inner workings of a system are not known, but based on trends and/or other relationships between data segments in past observed measurements, it can be predicted—within a (known) band of uncertainty—how the system under observation will behave in the nearby future. Such a model can be built on observing the system and looking at the relationship between different measurement values throughout time. See [7] for an example of a monitoring system that observes (patterns in behavior) of elderly people.

The establishment of phenomenological prediction models requires learning or 'knowledge discovery', a process that is described in [2]. This can require analyzing large amounts of historical sensor data. This is because—especially when little to knowledge is available on the system to be monitored—many sensor data needs to be taken into account, as well as long historical time series of sensor data. Also these models tend to be intertwined during their development phase with humans. During the learning phase, researchers and engineers will tend to look at different parameters by adding more (or less) sensors, apply different ways of interpretation (change algorithms), etc. In the case of dike LSMSs it turned out during research that it is sometimes very difficult to create a descriptive model, since dikes sometimes are from medieval times and no construction schematics are present. Little is known about the inner structure of these dikes, in this case the use of phenomenological models might be needed to determine structural behavior in the future, based on passed measurements.

While the establishment of a phenomenological model might require analysis of large amounts of data, once it has been established and the behavior of an observed system does no longer change, this is analysis is no longer needed. However, in practice the issue remains how to be sure that the behavior of the system under observation does not change. Not surprisingly, (end-)users of an LSMS tend to prefer to have a understanding of the systems for which they are responsible and thus prefer a **descriptive** model. A phenomenological model can sometimes help to deduce a descriptive model, by trying to build such a model that explains cause and effect relationships found in measured sensor data during the establishment of a phenomenological model. A detailed description of this kind of system modelling and learning is beyond the scope of this chapter.

3.1.3 Growth of Available Data

Next to the abstraction level of information and temporal issues, there is another Big Data aspect to monitoring, which is the growth of available data [8, 9]. Due to the advances in microelectronics, sensors can produce more data in a faster way. Data can be transported using more bandwidth and be stored in much larger quantities. While processing power also has increased, the growth of data remains an issue. Especially in the case of the establishment of phenomenological models—as described above—it is, in the beginning, often unknown what data should be measured and processed by the LSMS and what data is not relevant for producing the desired information. This results in the acquirement of a large amount of data, which all has to be processed.

Based on this general high level overview of several aspects of the relationship between Big Data and monitoring, it is possible to look at the relationship from three different constraint-based points of view.

3.2 Performance

The ability to act upon information (e.g. a warning) is key in the successful deployment of a LSMS. In this chapter the speed at which an LSMS can produce the desired information is defined as its performance. The amount of time between the moment a warning is issued and the last point in time the warning can be acted upon is defined in this chapter as the 'time window'. The larger the time window, the better the performance of an LSMS. For example, if it will take a year to carry out maintenance, than the monitoring system will have to warn at least a year ahead when it predicts failure 'in about a year'.

The size of a time window can be influenced by several factors. The MID decomposition is used to describe the relationship with Big Data from the performance constraint point of view.

Measurement. The amount of time involved in measurement is roughly determined by three things. First, the amount of time needed for measuring a physical aspect of the real world. Secondly, the amount of time needed for transporting the measurement to a location where interpretation of the data can take place. And finally, the amount of time needed for storing the information at a location, so it can be accessed by a data processing component. The total time involved largely depends on the amount of data measured and available bandwidth.

Note that the needed bandwidth and type of sensor are often closely intertwined. This relationship between a (visual) sensor and transport bandwidth is clearly shown by the example of a camera. Does it—for example—provide an HD movie stream of 25 images per second or does it act as an hourly snapshot camera? Or does the camera only send a new snapshot when something changes in the image?

Interpretation. The amount of time involved in interpretation roughly depends on three things. First, the amount of time needed for retrieving data from storage for processing. Then, the amount of time needed for processing data into information using (a) model(s) and finally the amount of time needed for storing information after processing. Note that if the data is processed in a streaming fashion instead of batch wise, less time might be needed.

Decision. The amount of time needed for making a decision to inform/warn (end-)users of an LSMS is roughly determined by two factors. First, the amount of time it takes for a rule set to be applied on the produced information. The amount of time it takes for a issued warning to reach the people or systems that can act on the information. Simply stated: the more time is needed for each of these steps, the less time remains for the warning time window.

3.3 Availability

Besides performance constraints, there are also availability constraints. A LSMS that is not available and does not provide a warning when it is needed is rather useless. It does not matter if the unavailability is due to expected monitoring systems maintenance or due to unexpected other causes. Note that the level of availability depends on the specific context of an LSMS. For example, if a system under observation can quickly change its behavior, the LSMS obviously needs a high level of availability, since else this change would be missed. However, in the case of slow changing systems, temporal non availability is allowed, because the time window (as described in the performance constraint viewpoint above) is relatively large.

The availability of an LSMS can be influenced by several factors. The MID decomposition is used to describe the relationship with Big Data from the availability constraint point of view.

Measurement. The availability of an LSMS with respect to measurement depends on the availability of the sensors, transport and storage. Especially storage is important from a Big Data point of view: this is where techniques for fast storage and retrieval of data come into play, as will be shown in the solution approach section later on.

Interpretation and Decision. The availability of the interpretation and decision parts depends on the availability of storage and computational facilities. Designers and engineers of an LSMS must take into account that these facilities can fail and be temporarily unavailable. This is where Big Data techniques can come into play with respect to redundancy, as will be shown later on.

3.4 Reliability

Even if the performance and availability of an LSMS are within the constraints set forward by the end-users in their context, an LSMS might still not be deployable because of reliability constraints. As described in the dike LSMS example in the monitoring section: not issuing a warning on structural dike failure is not acceptable. Also, sending out false alarms is not acceptable as people will no longer respond to alarms, even if this system is right at the time. In practice, reliability is closely related to availability, because the temporarily unavailability of components of a monitoring system can make the entire system less reliable.

The reliability of an LSMS can be influenced by several factors. The MID decomposition is used to describe the relationship with Big Data from the reliability constraint point of view.

Measurement. Reliability can be influenced by measurement errors, transport and storage errors. This means that data might get corrupted somehow.

Interpretation and decision. The reliability of the information produced by the LSMS directly relies on the data processing algorithms used. At the same time, the reliability is also influenced by the reliability of the computational and storage facilities (i.e. protection against corruption of data and information).

4 Solutions Within Constraints

In the previous sections monitoring systems have been generically described from a Big Data optimization point of view. Specific Big Data related requirements have been identified that have to be met in order for a monitoring system to be useful. In this section different solution approaches for meeting these constraints will be described.

For reasons of scope and size of this chapter, solution approaches concerning faster and bigger hardware will not be described, even though they help in certain situations. The focus will be on the data, algorithms and system architecture/design.

4.1 Approaches in Optimization

In the previous sections three types of constraints have been listed with respect to Big Data (optimization). First, the maximum size of the warning time window. Secondly, the availability of a monitoring system and finally the reliability of the warning itself. The relation with Big Data lies within the fact that the amount or availability of data involved can cause constraint violation. Optimization techniques can help avoid these violations.

In this section optimization techniques are categorized using several approaches in design and implementation of monitoring systems:

1. **Data oriented**. Deal with the amount of data by somehow reducing the volume, before it is processed by the data interpretation algorithms.
2. **Analysis oriented**. Enhance data interpretation algorithms, enabling them to deal with large amounts of data.
3. **System/architecture oriented**. Instead of reducing the amount of data or altering the data to information algorithms, the system as a whole is set up in such a way it can deal with large(r) amounts of data and/or in less time.

Note that designers can also try to resort to a fourth approach:

4. **Goal oriented**. Loosen the constraints. Only possible if there is still a business case for the system after loosening constraints.

Often, design and/or engineering solutions for keeping a monitoring system within constraints, cannot be positioned on a specific axis of orientation. For example, a solution might be data and algorithm oriented at the same time. In this chapter however, the distinction is made in order to position different types of solutions. Note that solution approaches will be described. Not the solutions themselves, since they require more detailed explanation.

4.1.1 Data Oriented Optimization

In general the amount of data to be processed can prevent a LSMS from having a small enough time window. A relatively simple optimization step is to reduce the volume of the (monitoring) data, resulting in a reduction of time needed in all process steps. In the case of the IJkdijk LSMS, it was learned that well maintained dikes—not being in a critical structural health condition—'change' relatively slow. A warning dike LSMS could therefor suffice with a sample rate of 5 min. Reduction of data involved can be performed in a number of different ways. Since this book is about Big Data optimization, increasing the speed of transport, retrieval and storage is not covered.

Reducing the size of the data often results in a reduction of processing time, since less data for each analysis step tends to take less time for analysis. An often used approach to reduce data is aggregation. Instead of having a sample each minute, 60 samples are aggregate—using an average—into a sample each hour. Note that this may come at a price with respect to the reliability constraint: if it is important that 'sudden spikes' in measurement data are to be analyzed too, using averaging might result in a loss of spikes in the data set.

Another way of reducing data for analysis is to convert data to another domain that can be processed just as efficiently. For example cyclic signals in actual measurements can sometimes be converted from the time domain into the frequency domain using Fourier transformations. There are also techniques to convert measurement data graphs automatically into a sequence of symbols. For example: segmenting the graph into sections of horizontal (A), riser (B) and falling (C) slopes, the graph can be converted into a sequence of A, B, C symbols. With non-cyclic signals this can result in a heavy data reduction. Note that again, this optimization technique can only be applied, if the LSMS as a whole stays within the reliability constraint: users must still be able to rely on the LSMS. An example of data reduction technique is called SAX and is described in [10].

A third data oriented optimization technique is to reorder the measurement data in such a way, that the analysis takes less time. An example case is the design of a cow LSMS in which the health and wellbeing of hundreds of thousands of cows is monitored throughout their lifetime [11]. This LSMS provides information on dairy production, which is needed by farmers to maximize the return on investment in milk production. At farms weight data is collected device centric, as depicted in Fig. 4. This means for example that there are a few scales (at each milking robot) that determine weight and many cows. Every time a scale is tread upon, the weight is digitally stored with a timestamp. The milking robot knows which cow (i.e. RFID tag) was being milked (and weighed) at a certain point in time. By combining the data from the milking robot with the scales, the weight of a cow at a certain point in time can be deduced. The interpretation part of the cow LSMS is cow-centric. It provides information per cow through time. By (also) storing weight data in a cow-centric way, the algorithm for interpretation has to carry out far less data retrieval, speeding up the analysis.

In general data oriented optimization techniques do not increase availability or reliability. In fact, it might even decrease reliability because information is lost. This is where another design and/or engineering degree of freedom comes into play: optimize the analysis.

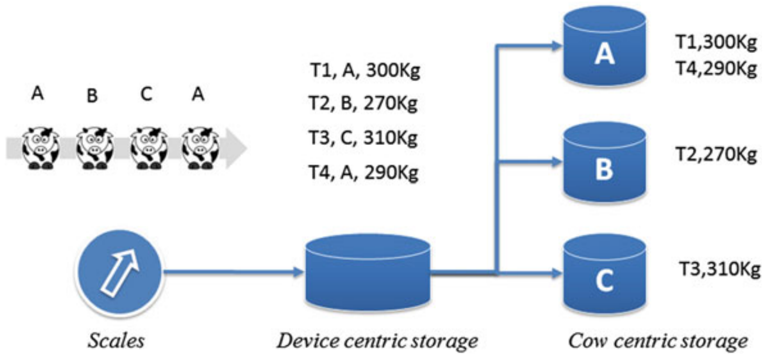

Fig. 4 Device centric and object centric storage

4.1.2 Analysis Oriented Optimization

During the analysis phase, data is converted into information using processing algorithms. The type of algorithms determines the reliability of the information and also the processing speed. So in order to meet time and reliability requirements, an approach is to focus on the algorithm at hand. Roughly stated, there are two kinds of analysis oriented approaches. The first approach is to optimize (or change) a specific algorithm (run on a single computer). This can influence the size of the time window or the level of reliability. A simple example of this approach—with respect to time constraints—is sorting a long list of measurement data. When using a ‘bubblesort’ algorithm, it takes for more time in general than using a ‘quicksort’ algorithm. Optimizing and/or designing improved data interpretation algorithms is very domain specific and outside the scope of this chapter.

Another well-known approach—with respect to the size of the time window—is to ‘divide-and-conquer’: carry out calculations for an algorithm in parallel. This can be done by creating a distributed algorithm or by separating data and run the same algorithm in parallel. Much Big Data optimization techniques revolve around some kind of ‘Map Reduce’ approach, where a larger problem is reduced into smaller one that can be handled separately [8, 9, 12, 13]. As stated before, this chapter is not about explaining these kind of solutions. They are multiple and require detailed explaining on their own.

4.1.3 System Architecture Oriented Optimization

Data oriented optimization and analysis oriented optimization are targeted at separate parts of the MID steps. It is also possible to optimize the monitoring system as a whole, across the ‘Measurement’ and ‘Interpretation’ part. In this section a number of them are reviewed.

More performance by distributing interpretation

By bringing the analysis towards the collection of data, it is sometimes possible to reduce the amount of transport, storage, retrieval and/or computation. An example can be found in [2]. This can be achieved by doing part of the analysis closer to the measurement points, reducing the need for transportation and central processing power (and time). Also, it is possible to carry out intelligent forwarding of information [7], where data or information is only forwarded when needed, by doing a small amount of interpretation close to the sensor, removing the need for a central point of interpretation to carry out all interpretation.

More performance with enough reliability by combined processing

Combining streaming and batch processing of data can also reduce the amount of processing time, while keeping reliability at acceptable levels. The idea is to have two separate and ‘parallel’ lines of processing data. The first processing line is targeted at speed: data that arrives is immediately processed and calculations are finished before the next data arrives. In this chapter this is defined as streaming data processing. Algorithms used for this type of processing often deal with a small portion of a time series, close to the most recent point in time (‘sliding window’). Warnings can be quickly issued, because processing takes place all the time. If the LSMS for some reason crashed, it can be ‘rebooted’ quickly again, thus increasing availability. However, because only a small portion of the historical behavior is taken into account, the monitoring system might miss certain long term trends. This might reduce the amount of reliability. This is where the second processing line comes into play. Next to the stream processing line there is also a batch processing line. It is targeted at processing more (historical) data in a single batch, but it requires more time. By using the batch processing line as a means to calibrate the stream processing line (e.g. ‘long term trends’), it is possible to get ‘the best of both worlds’.

The lambda architecture [14] is an example of combining processing lines. It states that the batch processing should be performed each time over all the data. To be able to produce also analyses over the latest data, a streaming system must be set up next to the batch which can processes the data in a streaming manner as long as the batch processing takes. At the moment the batch processing is finished, the results of the streaming analyses are overwritten by the batch results. The streaming analyses starts again, initialized with the newly batch results and the batch also is starting again.

Affordable performance through elasticity of resources

The analysis-oriented divide and conquer approach can require so much computational resources that the costs of storage and computation exceed the value of the information that the monitoring system produces. An approach to reduce the costs is to make use of resources in an elastic way.

The idea is to collect data at a relative low sample rate, which requires less storage and computational power for analysis. Once an unknown anomaly in the measurements is detected, the sample rate is increased and the anomaly can be analyzed using more sophisticated data processing algorithms, that are (temporarily) hosted on an elastic computing cloud. This is depicted in Fig. 5. If variation on this theme is doing a pre-analysis at the source of the data collections, instead of in the interpretation phase. An example can be found in an LSMS that monitors the impact of earthquakes in the Netherlands, which is currently under development. It is constructed to develop a deeper understanding of how vibrations of the underground impact houses. The quakes are caused by compacting of the subsurface at a depth of approximately 3 km, which in turn is the result of the decreasing of pore pressure in an underlying gas reservoir [15]. The LSMS design consists of hundreds of vibration sensors at several hundred houses. The structural health of the building is logged (e.g. ‘cracks in walls’) and by analyzing the combination of earthquake data, vibrations in houses and the structural health, researchers try to establish quantifiable cause and effect relationships between quakes and the resulting damage to a house. Since these types of quakes and impact on the houses are new, it is not known what for example the sample rate should be. Research and engineers currently thus want to collect data at a high sample rate. In order to keep the amount of processing as low as possible, the idea to reduce is to only forward high sample rate data if it is quake related. Only then computing power is needed for processing. This could be supplied by a

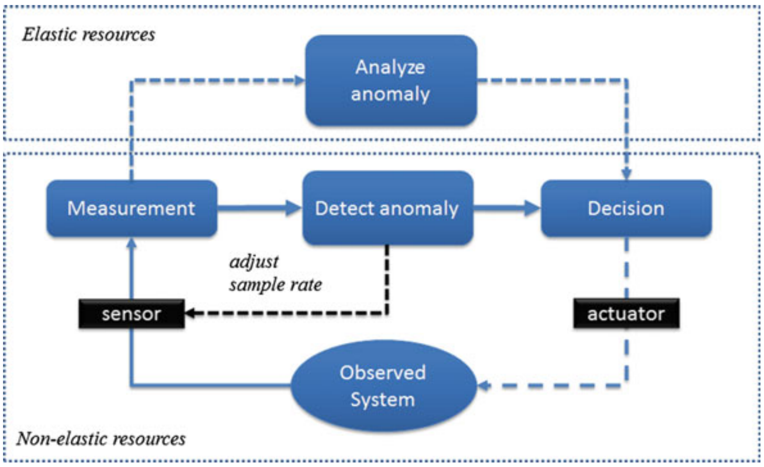

Fig. 5 Using elastic resources in monitoring

computational elastic cloud. After all quake data has been processed, the sample rate of forwarded data can be reduced again, which results in less computational power. More information on Big Data and elasticity can be found—for example—in [16].

Increasing availability through redundancy

As a final system architecture oriented optimization approach, the concept of adding redundancy is discussed. The basic idea is that failure of one of the components does not result in failure of the entire system ('no single point of failure'). By adding multiple components with the same functionality, some could fail, without a total failure of the entire LSMS. Describing general techniques for designing high availability (or sometimes called fault-tolerant systems) by using redundancy is beyond the scope of this chapter. Big Data computational platforms like Hadoop [17] and Project Storm [18] can provide—if configured properly—resilience against failing computational nodes. Also, NoSQL databases like Cassandra can store data with multiple copies of that data distributed over racks and even data centers if necessary.

An important aspect of increasing availability through redundancy is assuming everything can (and will) fail. Also, designers should create a system that tries to continue functioning, even if several components are not available anymore. The loss of a component should not result in a stop of the entire data to information processing chain. Instead, data and/or information should be rerouted to redundant components that still work. If loss of data is unacceptable (e.g. in a learning LSMS) data should only be allowed to be deleted in the monitoring chain, as long as there are at least two copies remaining elsewhere in the chain.

Increasing reliability through human error resilience

No matter how well tested data processing algorithms are, it is always possible a mistake has been made. Resulting in the production of unreliable information. One of the design goals of the Lambda architecture [14] discussed earlier is to be more robust against these kinds of human errors. LSMSs that are based on the Lambda architecture should preserve all original data and that it should be possible to reproduce al information using that data. Next to the streaming processing, there is the possibility to start a batch process that could use the entire dataset to reproduce the information. In this way the effects of a discovered error in a data processing algorithm can be mitigated to some extent.

4.1.4 Goal Oriented Optimization

As mentioned before, as a means of last resort, designers and engineers might loosen constraints. In a way, this means slightly altering the goal of the system. In practice designers and/or engineers sometimes discover not all of the original constraints from the specification do not have to be as tight as originally specified. By loosening constraints just a little, it is sometimes still possible to come up with a LSMS that still provides more value through information, than costs needed for realization and operation of an LSMS. For example, there are techniques—like the Early Accurate Result Library (EARL) system—which help determine the sample rate or the size of data samples, while staying with a specific error bound [19].

4.2 Summary of Optimizations and Constraints

As mentioned earlier on, there is no one-to-one mapping of constraints (time window, availability and reliability) and optimization techniques. In practice applying a specific technique often influences the achievement of several constraints. In the table below these relationships are once more illustrated and summarized, based on the descriptions above. A '+' means that the chance of staying within a constraint is increased, a '–' means that the chance is decreased. For example, aggregation of data can increase the chance of meeting a time window requirement, while decreasing the chance of meeting a reliability constraint. A '0' means little to no effect. Note that these are very generic scores, provided to show that improvements in the area of one constraint might result in issues with other constraints. In practice, the specific case and optimization technique at hand might lead to other values in the table (Table 1).

4.2.1 Impact on Other Constraints

Applying one or more Big Data optimization techniques as discussed in the previous paragraphs, can come at a price with respect to other constraints. The relationship between Big Data related constraints (performance, availability and reliability) have already been discussed. However, other constraints might be impacted too. This

Table 1 Relationship between optimations and constraints

	Performance	Reliability	Availability
Change data			
Aggregation	+	–	0
Concept conversion	+	0/−	0
Reordering	+	0	0
Change algorithms			
More efficient algorithm	+	0	0
Parallelization	++	0	0
Change architecture			
Faster: bring the analysis to the data	++	0	
Faster: combining streaming and batch	++	0	
Deal with many different sensor connections		0	++
Adapt resources according to need	0	0	–
Add redundant components	–	+	++
Create resilience against human error	0/−	+	0/+
Change goal			
Less perfect answers are acceptable	++	–	0

chapter would not be complete without mentioning several constraints that are important in modern day LSMS deployment too. This will be done in these last subsections.

Energy consumption

LSMS can consume large amounts of energy. Optimization techniques that require more hardware because of parallelization or redundancy, obviously tend to use more energy. Depending on the value of the information produced by the LSMS or the energy available in the deployment area of an LSMS, the energy usage might be more than allowed. The link between Big Data analytics and energy usage is covered in more detail in [20].

Maintainability

Intricate Big Data optimization techniques might enable designers and engineers of an LSMS to stay within the constrains as specified, but they could make it far more difficult for (other) designers and engineers to understand the LSMS as a whole. The ability to understand a system is also a constraint in everyday practice: without people that understand how the LSMS actually works it is difficult to maintain or improve the system. This understanding is needed, if for example unexpected errors take place and the LSMS needs some 'debugging' or a (partial) redesign. Even more so, if other people, which have not designed or implemented the system, have doubts about the reliability of the LSMS, it is very important that other experts—not involved in the realization of the LSMS—can get an understanding of the LSMS and provide the non-experts with a satisfactory answer to their doubts. In other words: maintainability through auditability is also a constraint, which can be influenced by Big Data optimization techniques in a negative way.

5 Conclusion

In this chapter the relationship between Big Data optimization and monitoring in the real world has been explored from different view-points. Several aspects of data collection and interpretation have been used as a viewpoint, as well as three types of constraints for large scale monitoring systems (abbreviated as LSMSs) that are related to Big Data: performance, availability and reliability constraints. Finally, several solution approaches involving Big Data optimization techniques have been provided that enable designers and engineers of LSMSs to provide a solution that stays within performance, availability and reliability constraints.

References

1. McCauley-Bell, P., Freeman, R.: Quantification of belief and knowledge systems in information warfare. In: Proceedings of the Fifth IEEE International Conference on Fuzzy Systems (1996). doi:10.1109/FUZZY.1996.552568
2. Chen, C.L.P., Zhang, C.-Y.: Data-intensive applications, challenges, techniques and technologies: a survey on big data. Inf. Sci. (2014). doi:10.1016/j.ins.2014.01.015
3. FloodControl IJkdijk. http://www.ijkdijk.nl/en/. Accessed 15 June 2015
4. Pals, N., De Vries, A., De Jong, A., Boertjes, E.: Remote and in situ sensing for dike monitoring: the Ijkdijk Experience. In: de Amicis, R., et al. (eds.) GeoSpatial Visual Analytics, pp 465–475. Springer, Netherlands (2009)
5. Roedelsperger, S., Coccia, A., Vicente, D., Trampuz, C., Meta, A.: The novel FastGBSAR sensor: deformation monitoring for dike failure prediction. In: 2013 Asia-Pacific Conference on Synthetic Aperture Radar (APSAR), pp. 420−423 (2013)
6. Helmholt, K., Courage, W.: Risk management in large scale underground infrastructures. In: 2013 IEEE International Systems Conference (SysCon) (2013). doi:10.1109/SysCon.2013.6549991
7. Jiang, P., Winkley, J., Zhao, C., Munnoch, R., Min, G., Yang, L.T.: An intelligent information forwarder for healthcare big data systems with distrib-uted wearable sensor. IEEE Syst. J. (2014). doi:10.1109/JSYST.2014.2308324
8. Kejariwal, A.: Big data challenges: a program optimization perspective. In: Second International Conference on Cloud and Green Computing (CGC) (2012). doi:10.1109/CGC.2012.17
9. Gu, L., Zeng, D., Li, P., Guo, S.: Cost Minimization for big data processing in geo-distributed data centers. IEEE Trans. Emerg. Top. Comput. (2014). doi:10.1109/TETC.2014.2310456
10. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A Symbolic representation of time series, with implications for streaming algorithms. In: DMKD '03 Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 2–11 (2003)
11. Vonder, M.R., Donker, G.: Cow-centric data made available in real-time for model development (2014). http://www.smartagrimatics.eu/Portals/4/SAM2014/1%20Farming%20in%20the%20Cloud/1.3%20Smart%20Livestock%20Farming/1.3.5%2020140618%20Agrimatics%202014%20-%20SDF%20%20InfoBroker%20v7.pdf. Accessed 15 June 2015
12. Ma, Y., Wang, L., Liu, D., Liu, P., Wang, J., Tao, J.: Generic parallel programming for massive remote sensing. In: 2012 IEEE International Conference on Data Processing (2012). doi:10.1109/CLUSTER.2012.51
13. Nita, M.-C., Chilipirea, Dobre, C., Pop, F.: A SLA-based method for big-data transfers with multi-criteria optimization constraints for IaaS. In: 2013 11th Roedunet International Conference (RoEduNet) (2013). doi:10.1109/RoEduNet.2013.6511740

14. Marz, N.: Lambda architecture. http://lambda-architecture.net. Accessed 15 June 2015
15. Ketelaar, G., van Leijen, F., Marinkovic, P., Hanssen, R.: Multi-track PS-InSAR datum connection. In: 2007 IEEE International Geoscience and Remote Sensing Symposium (IGARSS) (2007). doi:10.1109/IGARSS.2007.4423346
16. Han, R., Nie, L., Ghanem, M.M., Guo, Y.: Elastic algorithms for guaranteeing quality monotonicity in big data mining. In: 2013 IEEE International Conference on Big Data (2013). doi:10.1109/BigData.2013.6691553
17. Bicer, T., Wei Jiang, Agrawal, G.: Supporting fault tolerance in a data-intensive computing middleware. In: 2010 IEEE International Symposium on Parallel and Distributed Processing (IPDPS) (2010). doi:10.1109/IPDPS.2010.5470462
18. Chandarana, P., Vijayalakshmi, M.: Big data analytics frameworks. In: 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA) (2014). doi:10.1109/CSCITA.2014.6839299
19. Laptev, N., Zeng, K., Zaniolo, C.: Very fast estimation for result and ac-curacy of big data analytics: the EARL system. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE) (2013). doi:10.1109/ICDE.2013.6544928
20. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. J. Parallel Distributed Comput. (2014). doi:10.1016/j.jpdc.2014.01.003

Author Biographies

Kristian Helmholt is a Senior Consultant at TNO. After obtaining a master's degree in applied computing science, he worked on the creation of applications and services Internet Service Providers at the turn of the millennium. After that he started working on using sensor networks for data collection in different application domains like logistics and agriculture. By now he has specialized in the application of ICT for monitoring and control in asset management of large scale physical infrastructures, for example underground gas and electricity networks. A recurring theme in his work is the affordable and reliable acquisition of data which can be combined and refined into information for decision makers.

Bram van der Waaij is Senior Research Scientist at TNO. His department focuses on the monitoring and control of large scale infrastructures and how the many related organizations can use this insight in an optimal manner. He specializes in cloud based sensor data storage, Bigdata streaming analysis and multi-party collaboration. Research topics include cloud based analysis, column based databases and generic anomaly detection algorithms. All with a focus on inter organizational collaboration and multi-use of sensor systems. He applies his knowledge in various industry sectors, among others: dike management, smart grids and agriculture.